

openSUSE

www.suse.com

2012/09/09

システム分析とチューニングガイド



システム分析とチューニングガイド

Copyright © 2006–2012 Novell, Inc. and contributors. All rights reserved.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or (at your option) version 1.3; with the Invariant Section being this copyright notice and license. A copy of the license version 1.2 is included in the section entitled 「GNU Free Documentation License」.

For Novell trademarks, see the Novell Trademark and Service Mark list <http://www.novell.com/company/legal/trademarks/tmlist.html>. All other third party trademarks are the property of their respective owners. A trademark symbol (®), # etc.) denotes a Novell trademark; an asterisk (*) denotes a third party trademark.

All information found in this book has been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. Neither Novell, Inc., SUSE LINUX Products GmbH, the authors, nor the translators shall be held liable for possible errors or the consequences thereof.

下記に上記の日本語翻訳を掲載します。日本語の翻訳は公式なものではないことに注意してください。

Copyright © 2006–2012 Novell, Inc. および貢献者が全権利を留保しています。

この文書を、フリーソフトウェア財団発行の GNU フリー文書利用許諾契約書 バージョン 1.2 または (希望すれば) 1.3 が定める条件の下で複製、頒布、あるいは 改変することを許可する。ただし、この著作権とライセンス表記については変更不可部分とする。この利用許諾契約書の複製物は「GNU フリー文書利用許諾契約書」という章に含まれている。

Novell 社の商標については、Novell 社の商標とサービスマーカー覧 <http://www.novell.com/company/legal/trademarks/tmlist.html> をご覧ください。Linux は Linus Torvalds 氏による登録商標です。その他の商標は 各所有者の所有物です。商標シンボル (®, # など) は それぞれ Novell 社の商標であることを示しています。また、アスタリスク (*) は 第三者の商標を示しています。

この書籍内にある全ての情報は細部に至るまで最大限の注意を払って制作されていますが、完全に正確であることを保証するものではありません。Novell, Inc., SUSE LINUX Products GmbH, 著者, 翻訳者のいずれも、本書籍内の誤りとそこから生じる結果について、一切の保証はいたしません。

目次

このガイドについて	ix
パート I 基本	1
1 システムチューニングに対する一般的な注意事項	3
1.1 どの問題を解決すべきかの確認	3
1.2 一般的な問題の除外	4
1.3 ボトルネックの発見	5
1.4 一歩ずつのチューニング	5
パート II システム監視	7
2 システム監視ユーティリティ	9
2.1 多用途ツール	10
2.2 システム情報	17
2.3 プロセス	23
2.4 メモリ	28
2.5 ネットワーク	31
2.6 /proc ファイルシステム	34
2.7 ハードウェア情報	37
2.8 ファイルとファイルシステム	38
2.9 ユーザ情報	41
2.10 日付と時刻	42
2.11 データのグラフ化: RRDtool	42
3 Nagios を利用した監視	49
3.1 Nagios の機能	49
3.2 Nagios のインストール	49
3.3 Nagios の設定ファイル	50
3.4 Nagios の設定	53
3.5 トラブルシューティング	57

3.6	さらなる情報.	57
4	システムログファイルの分析と管理	59
4.1	/var/log/ 内にあるシステムログファイル.	59
4.2	ログファイルの閲覧と分析.	62
4.3	logrotate を利用したログファイルの管理.	63
4.4	logwatch を利用したログファイルの監視.	64
4.5	システムログへの記録コマンド logger.	65
パート III	カーネル監視	67
5	SystemTap—システムデータのフィルタと分析	69
5.1	考え方の概要.	69
5.2	インストールと設定.	72
5.3	スクリプトの文法.	74
5.4	スクリプト例.	82
5.5	さらなる情報.	83
6	カーネル探査	85
6.1	サポートされるアーキテクチャ.	86
6.2	カーネル探査の種類.	86
6.3	カーネルの探査 API.	87
6.4	Debugfs インターフェイス.	88
6.5	さらなる情報.	89
7	perfmon2—ハードウェアベースの性能監視	91
7.1	概念.	91
7.2	インストール.	93
7.3	perfmon の使用.	94
7.4	debugfs からの統計情報取得.	97
7.5	さらなる情報.	100
8	OProfile—システム全体に対するプロファイル	101
8.1	考え方の概要.	101
8.2	インストールと要件.	102
8.3	利用可能な OProfile ユーティリティ.	102
8.4	OProfile の使用.	103
8.5	OProfile GUI の使用.	105
8.6	レポートの作成.	106
8.7	さらなる情報.	107

パート IV	リソース管理	109
--------	--------	-----

9	一般的なシステムリソース管理	111
---	----------------	-----

9.1	インストールの計画	111
9.2	不要なサービスの無効化	113
9.3	ファイルシステムとディスクアクセス	114

10	カーネルのコントロールグループ	117
----	-----------------	-----

10.1	技術概要と用語定義	117
10.2	シナリオ	118
10.3	コントロールグループサブシステム	118
10.4	コントローラグループの使用	121
10.5	さらなる情報	124

11	電源管理	125
----	------	-----

11.1	CPU における電源管理	125
11.2	Linux カーネルの CPUfreq インフラストラクチャ	128
11.3	電源関係の設定の閲覧／管理／チューニング	131
11.4	特殊なチューニングオプション	137
11.5	電源管理プロファイルの作成と使用	139
11.6	トラブルシューティング	141
11.7	さらなる情報	142

パート V	カーネルのチューニング	143
-------	-------------	-----

12	複数のカーネルバージョンのインストール	145
----	---------------------	-----

12.1	複数バージョン対応の有効化と設定	146
12.2	YaST を利用した複数のカーネルバージョンのインストール／削除	148
12.3	zypper を利用した複数のカーネルバージョンのインストール／削除	149

13	I/O 性能のチューニング	151
----	---------------	-----

13.1	I/O スケジューラの切り替え	151
13.2	利用可能な I/O エレベータ	152
13.3	I/O バリアのチューニング	154

14	タスクスケジューラのチューニング	155
----	------------------	-----

14.1	前置き	155
14.2	プロセスの分類	157

14.3	O(1) スケジューラ	158
14.4	Completely Fair Scheduler (完全公平型スケジューラ)	159
14.5	さらなる情報	168
15	メモリ管理サブシステムのチューニング	171
15.1	メモリの使用方法	172
15.2	メモリ使用率の削減	174
15.3	仮想メモリマネージャ (VM) におけるチューニングパラメータ	175
15.4	不均一型メモリアクセス (Non-Uniform Memory Access (NUMA))	178
15.5	VM 動作の監視	179
16	ネットワークのチューニング	181
16.1	カーネルのソケットバッファの設定	181
16.2	ネットワークのボトルネック検出とネットワークトラフィックの分析	183
16.3	netfilter	184
16.4	さらなる情報	184
パート VI	システムダンプの処理	185
17	トレースツール	187
17.1	strace を利用したシステムコールのトレース	187
17.2	ltrace を利用したライブラリ呼び出しのトレース	191
17.3	Valgrind を利用したデバッグとプロファイル	192
17.4	さらなる情報	197
18	kexec と kdump	199
18.1	概要	199
18.2	必要なパッケージ	200
18.3	kexec の内側	200
18.4	基本的な kexec の使用方法	201
18.5	kexec を頻繁な再起動用に設定する方法	203
18.6	基本的な kdump の設定	203
18.7	クラッシュダンプの解析	207
18.8	高度な kdump 設定	212
18.9	さらなる情報	213
A	GNU ライセンス	215
A.1	GNU General Public License	215
A.2	GNU 一般公衆利用許諾契約書 (日本語訳)	219
A.3	GNU Free Documentation License	223

A.4 GNU フリー文書利用許諾契約書. 228

このガイドについて

openSUSE はエンタープライズ環境や科学データセンターなどの用途で幅広く 使用されています。SUSE は openSUSE を様々な用途に対して最適な 性能を発揮 するよう確信していますが、場合によっては openSUSE がファイル サーバではなく 複雑な計算を行なうサーバとして配置される場合など、全く異なる 要件に対応しな ければならない場合があります。

一般的には、すべての種類の処理に対して既定で最適化を行なって ディストリビューションを配布するのは不可能です。これは、全く異なるタイプの 処理に対しては、全く異なるタイプの要件が求められるためです。たとえば もっとも重要な I/O アクセスパターンやメモリアクセスのパターン、プロセスの スケジューリングなどがあ げられます。たとえ特定の処理に対して完全に適合する 設定を行なうことができたとしても、全く異なる処理に対しては通常、かえって 性能を悪化させてしまうことにな ります (たとえば I/O の集中するデータ ベースでは、ビデオエンコードなど CPU に処理が集中する場合に比べると、要件が 大きく異なります)。しかしながら、Linux の多用途性により、システムを それぞれの使用形態にあわせて設定すること ができるため、これによって最適な 性能を引き出すことができます。

このマニュアルでは、システム資源の監視方法やチューニング方法など、お使いの システムを監視したり分析したりする手順を紹介しています。なお、本ガイドでは 特定の用途に対して対処方法を示すもの *ではありません*。どちらかというとお使いの システムを大まかに分析し、性能を最大限に引き出す ための方法を紹介していま す。

システムチューニングに対する一般的な注意事項

システムのチューニング作業を実施するに当たっては、注意深く計画する必要 が あります。ここでは、お使いのシステムの性能改善を成功させるための必要 な 手順が示されています。

パートII「システム監視」(7 ページ)

Linux ではシステムにおけるほぼすべての要素を監視するため、数多くのツールが 提供されています。ここではこれらのユーティリティの使用方法や、システ ムログ ファイルの読み方などを記述しています。

パートIII「カーネル監視」(67 ページ)

Linux カーネル自身には、システムのパーツ一つ一つに対して調査を行なう機能 が 用意されています。このパートでは SystemTap と呼ばれる、各種のデー タ分析や フィルタを行なうためのスクリプト言語を紹介しています。スクリプト はカーネル モジュールに変換することができるため、カーネルのデバッグ情報

を収集すること ができるほか、探査機能を利用してボトルネックを発見したり、CPU の性能監視 ユニットにアクセスする `perfmon2` を利用したりすることができます。また、`Oprofile` を利用する監視アプリケーションについても紹介しています。

パートIV「リソース管理」(109 ページ)

サーバの正確な要件にあわせて、システムを仕立て上げるための方法を紹介しています。ここでは要件に適合するシステム性能を確保しながら、電源管理を行なう ための方法も紹介しています。

パートV「カーネルのチューニング」(143 ページ)

Linux カーネルは `sysctl` や `/proc` ファイルシステム を利用することで、最適化を行なうことができます。このパートでは、I/O 性能をチューニングしたり、Linux のプロセススケジューリングを最適化したり する方法について説明しています。また、ここではメモリ管理の仕組みについて 基本的な説明を行なっているほか、特定のアプリケーションや使用形態に 合わせてメモリ管理を最適化する方法についても紹介しています。それ以外にも、ネットワーク性能を改善するための方法も説明しています。

パートVI「システムダンプの処理」(185 ページ)

このパートでは、アプリケーションやシステムがクラッシュしたときの分析や 処理方法を紹介しています。`strace`、`ltrace` などのトレースツールのほか、`Kexec` や `Kdump` を使用するシステムクラッシュファイルの処理方法についても 説明しています。

また、このマニュアル内にある多くの章では、追加のドキュメンテーション資源に 対するリンクが書かれています。追加のドキュメンテーション資源はシステム上に 存在する場合のほか、インターネット上で公開されているものもあります。

お使いの製品に対して、利用可能なドキュメンテーションの概要や更新情報を 知るには、<http://www.suse.com/documentation> または 下記の章をお読みください:

1 利用可能な文書

HTML 版や PDF 版の各マニュアルは、それぞれ各種の言語に翻訳されています。この製品に対しては、それぞれ下記に示す ユーザ向けおよび管理者向けマニュアルが用意されています:

スタートアップ (↑ スタートアップ)

DVD や ISO イメージから `openSUSE` のインストールを行ない、`GNOME` や `KDE` デスクトップの簡単な説明と、そこで動作する主なアプリケーションを紹介

するまでの範囲を説明しています。また、LibreOffice の概要説明のほか、文書作成や表計算での作業、およびグラフィックやプレゼンテーションの作成を行なうためのモジュールについても説明しています。

リファレンス (↑リファレンス)

openSUSE に関する一般的な理解を深め、より詳しいシステム管理作業を行なうための情報が書かれています。主にシステム管理者のほか、システム管理知識のあるホームユーザに向けた文書です。また、複雑な配置シナリオやシステムの管理方法、主なシステムコンポーネントとのやりとりや openSUSE が提供するネットワークサービス、ファイルサービスに関する詳しい情報も書かれています。

セキュリティガイド (↑セキュリティガイド)

ローカル環境やネットワークセキュリティを含めた、システムセキュリティに関する基本的な考え方が書かれています。AppArmor のようなセキュリティソフトウェア (プログラムが読み書きしたり実行したりするファイルをプログラム単位で指定できるもの) の一般的な使い方を示しているほか、セキュリティ関連のイベント情報を確実に収集するための監査システムの使い方も示しています。

システム分析とチューニングガイド (i ページ)

問題の検出や解決、最適化に対する管理者向けのガイドです。お使いのシステムに関して監視ツールを利用し点検と最適化を行なう方法や、効率的に資源を管理するための手順が記されています。また、一般的によくある問題やそれに対する解決方法、追加のヘルプや文書資源についても示しています。

KVM を利用した仮想化 (↑KVM を利用した仮想化)

このマニュアルでは、openSUSE で KVM (カーネルベースの仮想マシン) による仮想化を設定したり、管理したりするための手順を紹介しています。また、libvirt や QEMU を利用した VM ゲストの管理方法についても紹介しています。

ほとんどの製品マニュアルは HTML 版の形で、インストール済みシステムの `/usr/share/doc/manual` に置かれています。またデスクトップのヘルプセンターからもアクセスすることができます。最新の文書は、<http://www.suse.com/documentation> に置いています。ここからお使いの製品について、PDF 版と HTML 版をダウンロードすることができます。

2 フィードバック

いくつかの方法でフィードバックを送ることができます:

バグや機能追加リクエスト

製品のコンポーネントに対してバグの報告を行なったり、もしくは機能の追加リクエストを送信したりしたい場合は、<https://bugzilla.novell.com/> をご利用ください。文書内の間違いについては、各製品の *Documentation* コンポーネントに対してバグ報告をお願いいたします。

Bugzilla を初めてお使いになる場合は、下記の記事をお読みください:

- http://ja.opensuse.org/Submitting_bug_reports
- http://ja.opensuse.org/Bug_reporting_FAQ

ユーザコメント

このマニュアルに対するコメントや提案のほか、この製品に含まれる他のドキュメント 類に対するコメントを歓迎します。オンラインドキュメントの場合は、それぞれの ページ下部にあるコメント機能をご利用いただくか、もしくは <http://www.suse.com/documentation/feedback.html> から コメントをお送りください。

メール

この製品に対するフィードバックを送信するには、doc-team@suse.de 宛のメールもお使いいただけます。それぞれドキュメントのタイトルと製品バージョン、発行日時を添えてお送りください。また、間違いの報告や加筆に対する提案につきましては、その簡潔な説明と、セクション番号およびページ (または URL) をお送りください。

3 文書規約

このマニュアルでは、下記のルールで文書を記述しています:

- `/etc/passwd`: ディレクトリ名やファイル名を示しています
- `placeholder`: 置き換えを示しています `placeholder` を実際の値に置き換えます
- `PATH`: `PATH` という名前の環境変数を示しています
- `ls, --help`: コマンドやオプション、パラメータ を示しています
- `user`: ユーザまたはグループ

- , + F1: 入力するキーやキーの組み合わせを示しています; キーはキーボードに書かれているとおりに大文字で示されます
- ファイル, ファイル > 名前を付けて保存: メニュー項目やボタンなどを示しています
- ダンシングペンギン (他のマニュアル内 ペンギン の章): 他のマニュアル内にある章を示しています

4 このマニュアルの作成について

この書籍は、DocBook (詳しくは <http://www.docbook.org> をご覧ください) のサブセットである Novdoc で書かれています。XML のソースファイルは xmllint で検証された後に xsltproc で処理され、Norman Walsh 氏のスタイルシートのカスタマイズ版を利用して XSL-FO に変換されます。最終的な PDF ファイルは RenderX 提供の XEP で生成しています。また、この マニュアルを構築するために使用するオープンソースツールとその環境は、openSUSE と共に公開されている daps パッケージ内にあります。なお、daps の Web ページは <http://daps.sf.net/> です。

5 ソースコード

openSUSE のソースコードは、どなたにでもご利用いただけます。ダウンロードのリンクやその他の説明については、http://ja.opensuse.org/Source_code をお読みください。

6 謝辞

多数の無償貢献のお陰で、Linux 開発者はその開発にあたってグローバルな協力を行なうことができています。我々は彼らのそのような努力に感謝します— 彼らの貢献がなければ本ディストリビューションは存在していませんでした。また、Frank Zappa 氏と Pawar 氏にも感謝しています。もちろん Linus Torvalds 氏には特に感謝しています。

Have a lot of fun!

SUSE チームより

パートⅠ. 基本

システムチューニングに対する一般的な注意事項

このマニュアルでは、性能問題に対する原因の追及方法のほか、これらの問題を解決するための方法を説明しています。なお、お使いのシステムに対してチューニングを始める前に、一般的な問題を除外できているかどうかを確認し、問題の根本原因（ボトルネック）を見つけ出す必要があります。またこれ以外にも、システムのチューニングについて、詳細な計画を立てる必要もあります。これは手当たり次第にチューニングを行なってしまうと問題の解決に至らないばかりか、かえって問題を悪化させてしまう場合があります。

手順 1.1 システムチューニング時の一般的なアプローチ

- 1 どの問題を解決すべきかを確認する。
- 2 一般的な問題を除外する。
- 3 ボトルネックを見つける。
 - 3a システムやアプリケーションを監視する。
 - 3b データを分析する。
- 4 一歩ずつチューニングを行なう。

1.1 どの問題を解決すべきかの確認

お塚のシステムに対してチューニングを行なう前に、まずは問題点をできる限り詳細に説明できるように努めてください。言うまでもないことですが、シンプルで一

般的すぎる「システムが遅い!」だけでは問題の説明にはなりません。たとえば、Web サーバについて静的なページをより高速に配信したい場合であれば、単純に全体的な速度を向上する必要があるのか、もしくはピーク時間帯の改善をはかりたいのかで話が変わってきます。

さらに言えば、ご自身が直面している問題に対して、具体的なデータ測定を行なうことができるかどうかを検討する必要もあります。データ測定を行なわないと、チューニングが成功したのかどうかの判断が付かなくなってしまうためです。また、チューニングの「実施前」と「実施後」での比較も重要です。

1.2 一般的な問題の除外

性能面の問題では、しばしばネットワークやハードウェアの問題やバグ、設定問題などが原因となる場合があります。システムのチューニングに取りかかる前に、まずは下記に示されたような事象が発生していないかどうかを確認し、一般的な問題を除外してください:

- まずは `/var/log/warn` や `/var/log/messages` を確認し、通常時は見られないようなメッセージが記録されていないかどうかを確認します。
- `top` コマンドや `ps` コマンドを利用し、CPU 時間やメモリを大幅に消費していたり、占有していたりするプロセスが存在しないかどうかを確認します。
- `/proc/net/dev` の内容を確認し、ネットワーク問題が発生していないかどうかを確認します。
- 物理的なディスクに I/O 問題が発生しているような場合は、これがハードウェア側の原因やディスク全体の原因でないことを確認します (たとえば `smartmontools` によるディスクチェックなどで確認)。
- バックグラウンドのジョブでは、サーバの負荷が低くなる時間帯にスケジュールされていることを確認します。また、これらのジョブは優先度の低い設定で動作させる必要があります (`nice` コマンドを使用します)。
- 同一の資源を利用する複数のサービスが動作しているマシンの場合は、他のサーバにサービスを移設できないかどうかを検討します。
- 最後に、お使いのソフトウェアが最新の状態であることを確認します。

1.3 ボトルネックの発見

システムをチューニングするに当たって、もっとも難しいのがボトルネックの発見です。openSUSE では、このような作業を行なうに当たって多数の ツールを提供しています。これらのシステム監視アプリケーションやログファイルの 分析について、詳しくは パートII「システム監視」(7 ページ) をお読みください。問題の原因を追及するのに長時間かつ深い分析を必要とする場合は、そのようなアプリケーションも用意されています。詳しくは パートIII「カーネル監視」(67 ページ) をお読みください。

データの収集を行なった後は、それらを分析する必要があります。まずはサーバのハードウェア情報 (メモリ, CPU, バス) とその I/O 性能 (ディスク, ネットワーク) を分析してみてください。これらの基本要件が正しく満たされていれば、システムをチューニングするための準備ができたものと言えます。

1.4 一歩ずつのチューニング

チューニングの実施は注意して行なってください。同時に複数の対応を実施したりせず、1 つずつ確実に実施してください。それぞれの変更によって改善されたのかどうかを確認するには、1 つずつ実施してそれぞれで測定を行なう必要があるためです。また、それぞれ実施したチューニングに対しては、十分な時間を確保してデータの測定を行ない、十分なデータ量に対して分析を行なってください。なお、改善が見られないような場合は、それらの変更は恒久的に 反映しないでください。変更によって将来、何らかの悪い影響を与えてしまう 場合が考えられるためです。

パート II. システム監視

システム監視ユーティリティ

お使いのシステムの状態を確認するため、数多くのプログラムやツール、ユーティリティなどが用意されています。この章では、これらユーティリティのうちのいくつかを紹介するほか、日々の定型作業をこなすのに便利なユーティリティや、それらの中の最も重要なパラメータについても紹介しています。

本章でコマンドを紹介する際には、コマンドと共にその出力例も表示しています。たとえば下記の例では、最初の行がコマンドそれ自身（行頭に `>` や `#` が書かれます）を示し、それ以降がコマンドの出力例になっています。途中の出力などを省略する場合は、大括弧 (`[...]`) を利用して表記します。また、表記上の理由で長い行を途中で改行する場合は、バックスラッシュ（日本語環境では円マーク）（`\`）を行末に表記します。

```
# command -x -y
出力 行 1
出力 行 2
出力 行 3 は厄介なまでに長いので、改行を入れて \
        行を区切っています。
出力 行 4
[... ]
出力 行 98
出力 行 99
```

説明は、できる限り多くのユーティリティを紹介したいという意図から、できる限り簡潔に記しています。全てのコマンドに対する詳しい情報は、各マニュアルページをお読みください。多くのコマンドでは `--help` オプションを指定することにより、利用可能なパラメータの一覧と、簡易な説明が表示されます。

2.1 多用途ツール

多くの Linux システム監視ツールは、システムの一部を監視するのに特化した仕組みになっていますが、「十徳ナイフ」のようなシステム状態を一括表示 するようなものも少数存在しています。まずはこれらのツールを利用してシステム状態の 概要をつかむとともに、どの部分についてより細かい調査を必要とするのかを確認してください。

2.1.1 vmstat

vmstat はプロセスやメモリ、I/O や割り込み、CPU に関する情報を収集します。パラメータに採取間隔を指定せずに実行すると、直近の再起動以後の平均値を表示します。パラメータに採取間隔を指定して実行すると、実際の採取間隔ごとに値を表示します：

例 2.1 vmstat 負荷の軽いマシンでの出力

```
tux@mercury:~$ vmstat -a 2
procs -----memory----- ---swap-- ---io--- -system-- -----cpu-----
 r  b   swpd   free  inact active    si   so    bi   bo   in  cs us sy  id wa st
 0  0     0 750992 570648 548848    0    0    0    1    8   9  0  0 100  0  0
 0  0     0 750984 570648 548912    0    0    0    0   63  48  1  0 99  0  0
 0  0     0 751000 570648 548912    0    0    0    0  55  47  0  0 100  0  0
 0  0     0 751000 570648 548912    0    0    0    0  56  50  0  0 100  0  0
 0  0     0 751016 570648 548944    0    0    0    0  57  50  0  0 100  0  0
```

例 2.2 vmstat 負荷の重いマシンでの出力 (特に CPU を使用しているマシン)

```
tux@mercury:~$ vmstat 2
procs -----memory----- ---swap-- ---io--- -system-- -----cpu-----
 r  b   swpd   free  buff  cache    si   so    bi   bo   in  cs us sy  id wa st
32  1 26236 459640 110240 6312648    0    0 9944    2 4552 6597 95  5  0  0  0
23  1 26236 396728 110336 6136224    0    0 9588    0 4468 6273 94  6  0  0  0
35  0 26236 554920 110508 6166508    0    0 7684 27992 4474 4700 95  5  0  0  0
28  0 26236 518184 110516 6039996    0    0 10830    4 4446 4670 94  6  0  0  0
21  5 26236 716468 110684 6074872    0    0 8734 20534 4512 4061 96  4  0  0  0
```

ヒント

vmstat の出力のうち最初の行には、常に直近の再起動からの平均値が表示されます。

各列には、それぞれ下記の値が表示されます：

r

実行キューに入っているプロセス数が表示されます。これらのプロセスは、処理を行なうために CPU の空きを待っている状態です。ここの値が利用可能な CPU 数 よりも定常的に大きい場合は、CPU の処理能力が不足していることを示します。

b

CPU 以外の資源を待機しているプロセス数が表示されます。ここの値が高い場合は、入出力 (ネットワークまたはディスク) に問題があることを示しています。

swpd

現在使用中のスワップ領域の量 (キロバイト単位) が表示されます。

free

未使用のメモリ量 (キロバイト単位) が表示されます。

inact

未使用メモリのうち、再利用可能な量の最新値を示しています。この列は、vmstat のパラメータに -a (推奨パラメータ) を指定した場合にのみ表示されます。

active

通常は再利用できない、直近で使用されているメモリの量を表示しています。この列は、vmstat のパラメータに -a (推奨パラメータ) を指定した場合にのみ表示されます。

buff

RAM 内にあるファイルバッファキャッシュのサイズ (キロバイト単位) が表示されます。この値は、vmstat のパラメータに -a (推奨パラメータ) を指定した場合には表示されません。

cache

RAM 内にあるページキャッシュのサイズ (キロバイト単位) が表示されます。この値は、vmstat のパラメータに -a (推奨パラメータ) を指定した場合には表示されません。

si

1 秒あたりにスワップ領域から RAM に移動したデータの量 (キロバイト単位) が表示されます。この列の値が長い時間高い値を示している場合、お使いのマシンに RAM を追加したほうがよいことを示しています。

so

1 秒あたりに RAM からスワップ領域に移動したデータの量 (キロバイト単位) が表示されます。この列の値が長い時間高い値を示している場合、お使いのマシンに RAM を追加したほうがよいことを示しています。

bi

1 秒あたりにブロックデバイス (ディスクなど) から読み出したブロック数が表示されます。スワップ処理が起こった場合にも、この値が増加します。

bo

1 秒あたりにブロックデバイス (ディスクなど) に書き込んだブロック数が表示されます。スワップ処理が起こった場合にも、この値が増加します。

in

1 秒あたりの割り込み数が表示されます。この値が高い場合、入出力 (ネットワークまたはディスク) が多く発生していることを示します。

cs

1 秒あたりのコンテキストスイッチ数が表示されます。この値を簡単に言うと、カーネルが複数のプログラムを実行するため、一方のプログラムから他方のプログラムに実行コードを切り替えた回数を示しています。

us

ユーザプロセスによる CPU 使用率をパーセンテージで示しています。

sy

システムプロセスによる CPU 使用率をパーセンテージで示しています。

id

CPU がアイドル状態 (未使用の状態) にあった割合をパーセンテージで示しています。この値が長い時間ずっとゼロであった場合、お使いの CPU が使い切られていることを示しています。これは特に悪い兆候というわけではありません。*r* や *b* の値をそれぞれ確認し、十分な CPU 性能があることを確認できれば、特にこの値が小さくても心配する必要はありません。

wa

"wa" の時間がゼロより大きい値を示している場合、データの入出力待ちでスルー プット (処理速度) が落ちていることを示しています。これはたとえばファイルを初めて読み込むような場合や、裏での書き込み処理が追いつかないような場合には避けられないものではありません。また、この値はハードウェア上のボトルネック (たとえばネットワークやハードディスク) が存在していることを示している場合もあります。それ以外にも、仮想メモリマネージャ (詳しくは 第15章

メモリ管理サブシステムのチューニング(171 ページ)をお読みください) のチューニングを行なう余地があることを示している場合もあります。

st

仮想マシンで使用された CPU 時間のパーセンテージを表示しています。

さらなるオプションについては、`vmstat --help` をお読みください。

2.1.2 システムの動作情報: `sar` と `sadc`

`sar` コマンドはほぼ全ての重要なシステム動作についてレポートを生成することのできるツールで、CPU やメモリ、IRQ の使用率、入出力やネットワークなどの情報を表示することができます。また本コマンドはレポートをその場で作成することができますほか、システム動作の収集プログラムである `sadc` が過去に採取した既存の情報から作成することもできます。なお、`sar` や `sadc` は全て `/proc` ファイルシステムが提供する情報を利用しています。

注記: `sysstat` パッケージ

`sar` と `sadc` は、いずれも `sysstat` パッケージに含まれています。YaST からインストールを行なうか、もしくは `zypper in sysstat` でインストールを行なってください。

2.1.2.1 `sadc` を利用した日次の自動データ収集

お使いのシステムを長い期間にわたって監視したい場合は、`sadc` を利用してデータを自動収集させることをお勧めします。収集したデータは後から `sar` コマンドで読み込むことができます。`sadc` を起動するには、`/etc/init.d/boot.sysstat start` を実行してください。このコマンドは下記の既定の設定で `sadc` を呼び出すよう `/etc/cron.d/` にリンクを追加するものです。

- 利用可能な全てのデータを収集する。
- データは `/var/log/sa/saDD` に書き込む。ここで `DD` は現在の日 (日付) を示すものとする。既にファイルが存在する場合、ファイルはアーカイブされる。
- 概要レポートは `/var/log/sa/sarDD` に書き込む。ここで `DD` は現在の日 (日付) を示すものとする。既にファイルが存在する場合、ファイルはアーカイブされる。
- データは 10 分間隔で収集し、概要レポートは 6 時間ごとに作成する (`/etc/sysstat/sysstat.cron` をお読みください)。

- データは /usr/lib64/sa/sa1 で収集する (32 ビット環境の場合は /usr/lib/sa/sa1)
- 概要は /usr/lib64/sa/sa2 で作成する (32 ビット環境の場合は /usr/lib/sa/sa2)

設定をカスタマイズしたい場合は、sa1 と sa2 のスクリプトをコピーし、必要に応じて修正してください。また、/etc/cron.d/sysstat にある シンボリックリンクを /etc/sysstat/sysstat.cron のカスタマイズ版と入れ替えてください。

2.1.2.2 sar を利用したレポート生成

レポートをその場で作成するには、sar コマンドに間隔 (秒) とカウントを指定して実行します。ファイルからレポートを作成するには、間隔指定の代わりに -f オプションを利用し、ファイル名を 指定してください。ファイル名と間隔、カウントのいずれも指定しなければ、/var/log/sa/sa~~DD~~ ファイルからレポートを生成します。ここで ~~DD~~ には今日の日 (日付) が入ります。上記のディレクトリは sadc がデータを書き込む際の既定のディレクトリです。複数のファイルを読み込む場合は、-f オプションを必要な数だけ指定してください。

```
sar 2 10 # その場でレポートを作成する。 2 秒間隔で 10 回分。
sar -f ~/reports/sar_2010_05_03 # sar_2010_05_03 ファイルへの問い合わせ。
sar # /var/log/sa/ にある本日分のファイルへの問い合わせ。
cd /var/log/sa &&¥
sar -f sa01 -f sa02 # /var/log/sa/0[12] のファイルへの問い合わせ。
```

sar の便利な使用例と解釈は後述します。各列の意味について、詳しくは sar の man (1) (マニュアルページ) をお読みください。マニュアルページには、sar が提供する数多くのオプション指定やレポート生成方法が書かれています。

CPU 使用率レポート: sar

何もオプションを指定せずに呼び出した場合、sar は CPU について基本的なレポートを作成します。マルチプロセッサマシンの場合、全ての CPU に対する結果が合計されて表示されます。個別の CPU に対して別々の統計 情報を得たい場合は、-P ALL オプションをお使いください。

```
mercury:~ # sar 10 5
Linux 2.6.31.12-0.2-default (mercury) 03/05/10 _x86_64_ (2 CPU)
```

	CPU	%user	%nice	%system	%iowait	%steal	%idle
14:15:43	all	38.55	0.00	6.10	0.10	0.00	55.25
14:16:03	all	12.59	0.00	4.90	0.33	0.00	82.18
14:16:13	all	56.59	0.00	8.16	0.44	0.00	34.81
14:16:23	all	58.45	0.00	3.00	0.00	0.00	38.55

14:16:33	all	86.46	0.00	4.70	0.00	0.00	8.85
Average:	all	49.94	0.00	5.38	0.18	0.00	44.50

%iowait (入出力を待機していた CPU 時間の割合) が長い 時間ゼロよりも顕著に大きい場合は、入出力のシステム (ネットワークやハード ディスク) にボトルネックが存在することを示しています。また、*%idle* が長い時間ゼロであった場合は、お使いの CPU の 性能を使い切っていることを示しています。capacity.

メモリ使用率レポート: sar -r

お使いのシステムメモリ (RAM) について概要を生成するには、*-r* オプションを使用します:

```
mercury:~ # sar -r 10 5
Linux 2.6.31.12-0.2-default (mercury) 03/05/10   _x86_64_   (2 CPU)

16:12:12 kbmemfree kbmemused %memused kbbuffers kbcached kbcommit %commit
16:12:22    548188    1507488    73.33    20524    64204    2338284    65.10
16:12:32    259320    1796356    87.39    20808    72660    2229080    62.06
16:12:42    381096    1674580    81.46    21084    75460    2328192    64.82
16:12:52    642668    1413008    68.74    21392    81212    1938820    53.98
16:13:02    311984    1743692    84.82    21712    84040    2212024    61.58
Average:    428651    1627025    79.15    21104    75515    2209280    61.51
```

最後の 2 列 (*kbcommit* と *%commit*) には、現在の負荷量进行处理にあたって必要な合計のメモリ容量 (RAM とスワップ の合計) の最大値をそれぞれキロバイト単位とパーセント単位で表示しています。

ページング統計レポート: sar -B

カーネルのページングに関する統計情報を表示するには、*-B* オプションを使用します。

```
mercury:~ # sar -B 10 5
Linux 2.6.31.12-0.2-default (mercury) 03/05/10   _x86_64_   (2 CPU)

16:11:43 pgpgin/s pgpgout/s  fault/s majflt/s  pgfree/s pgscank/s  pgscand/s pgsteal/s  %vmeff
16:11:53    225.20    104.00  91993.90      0.00  87572.60      0.00      0.00      0.00      0.00
16:12:03    718.32    601.00  82612.01      2.20  99785.69    560.56    839.24   1132.23    80.89
16:12:13   1222.00   1672.40 103126.00      1.70 106529.00  1136.00    982.40   1172.20    55.33
16:12:23    112.18     77.84  113406.59      0.10  97581.24    35.13   127.74   159.38    97.86
16:12:33     817.22     81.28 121312.91      9.41 111442.44      0.00      0.00      0.00      0.00
Average:    618.72    507.20 102494.86      2.68 100578.98   346.24   389.76   492.60    66.93
```

majflt/s (1 秒あたりのメジャーフォルト) の列には、ディスク (スワップ) からメモリ無いに読み込まれたページ数が書かれています。メジャーフォルトの値が大きくなるとシステムの速度を低下させてしまうため、メインメモリの不足を意味する指標になります。また、*%vmeff* の列にはメインメモリ内のキャッシュやスワップキャッシュ (*pgsteal/s*) に対する スキャン済みページ数 (*pgscand/s*) の割合が表示されてい

ます。これはページの 再利用について効率値を示すもので、100 に近い値を示す (それぞれスワップアウト された未使用のページが再利用されている) か、もしくは 0 に近い値を示す (ページがスキャンされていない) ほどシステムが効率よく働いていることを示します。なお、この値は 30 より小さい値になるべきではありません。

ブロックデバイスの統計情報: sar -d

ブロックデバイス (ハードディスク、光学ドライブ、USB ストレージデバイスなど) についての統計情報を表示するには、-d オプションを使用します。DEV の列を見やすくするため、-p (pretty-print) の追加オプションを使用していることをご確認ください。

```
mercury:~ # sar -d -p 10 5
Linux 2.6.31.12-0.2-default (neo) 03/05/10 _x86_64_ (2 CPU)

16:28:31 DEV      tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz  await  svctm  %util
16:28:41 sdc      11.51    98.50    653.45    65.32     0.10    8.83   4.87   5.61
16:28:41 scd0     0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00

16:28:41 DEV      tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz  await  svctm  %util
16:28:51 sdc      15.38   329.27   465.93    51.69     0.10    6.39   4.70   7.23
16:28:51 scd0     0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00

16:28:51 DEV      tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz  await  svctm  %util
16:29:01 sdc      32.47   876.72   647.35    46.94     0.33   10.20   3.67  11.91
16:29:01 scd0     0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00

16:29:01 DEV      tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz  await  svctm  %util
16:29:11 sdc      48.75  2852.45   366.77    66.04     0.82   16.93   4.91  23.94
16:29:11 scd0     0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00

16:29:11 DEV      tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz  await  svctm  %util
16:29:21 sdc      13.20   362.40   412.00    58.67     0.16   12.03   6.09   8.04
16:29:21 scd0     0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00

Average: DEV      tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz  await  svctm  %util
Average: sdc      24.26   903.52   509.12    58.23     0.30   12.49   4.68  11.34
Average: scd0     0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
```

お使いのマシンに複数のディスクが接続されている場合、全てのディスクに対して 平均的に入出力要求が分散していると最高の性能を引き出すことができます。Average 行の tps, rd_sec/s, wr_sec/s の値を 全てのディスクに対して比較してください。svctm と %util の列にある値が定常的に高い場合は、ディスク内の 空き容量が不足していることが考えられます。

ネットワークの統計情報: sar -n キーワード

-n オプションを指定すると、ネットワークに関連する複数の レポートを生成することができます。-n に続いて下記の キーワードを指定してください:

- *DEV*: 全てのネットワークデバイスに対する統計 レポートを生成します。
- *EDEV*: 全てのネットワークデバイスに対するエラー 統計レポートを生成します。
- *NFS*: NFS の統計レポートを生成します。client
- *NFSD*: NFS サーバの統計レポートを生成します。
- *SOCK*: ソケットに関する統計レポートを生成します。
- *ALL*: 全てのネットワーク統計情報を生成します。

2.1.2.3 sar データの視覚化

sar のレポートは人間にとってそれほど見やすいものとは言えません。kSar と呼ばれる sar データの視覚化 Java アプリケーションがありますが、これを利用することで見やすいグラフを作成することができます。また、このアプリケーションは PDF 形式でのレポートを作成することもできるほか、その場での作成や過去データのファイルから作成することもできます。なお、kSar は BSD ライセンスで提供されています。詳しくは <http://ksar.atomique.net/> をお読みください。

2.2 システム情報

2.2.1 デバイスの負荷情報: iostat

iostat はシステムデバイスの負荷を監視します。お使いのシステムに接続された物理ディスクの負荷について、うまく バランスをとるための便利なレポートを提供します。

最初の iostat レポートでは、システム起動後から 収集し続けられた統計情報を表示します。それ以降のレポートでは、前回の レポートからの差分を表示します。

```
tux@mercury:~$ iostat
Linux 2.6.32.7-0.2-default (geeko@buildhost) 02/24/10 _x86_64_

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0,49    0,01    0,10    0,31    0,00   99,09

Device:            tps    Blk_read/s    Blk_wrtn/s    Blk_read    Blk_wrtn
```

sda	1,34	5,59	25,37	1459766	6629160
sda1	0,00	0,01	0,00	1519	0
sda2	0,87	5,11	17,83	1335365	4658152
sda3	0,47	0,47	7,54	122578	1971008

-n オプションを指定すると、iostat はネットワークファイルシステム (NFS) に関する負荷情報を表示するようになります。また、-x オプションを指定すると、より広い範囲の統計情報を表示します。

また、どのデバイスをどれくらいの時間間隔で監視するのかを指定することができます。たとえば `iostat -p sda 3 5` のように入力すると、デバイス sda に関するレポートを 3 秒間隔で 5 回表示します。

注記: sysstat パッケージ

iostat は sysstat パッケージの一部です。YaST を利用するか、もしくは `zypper in sysstat` コマンドでパッケージをインストールしてください。

2.2.2 プロセッサの動作監視: mpstat

mpstat ユーティリティは、利用可能な各プロセッサの動作状況を調査します。お使いのシステムにプロセッサが 1 基しか搭載されていない場合は、全体の平均統計情報が表示されます。

-P オプションを指定すると、報告すべきプロセッサを番号で指定することができます (0 が 1 基目になります)。タイミングに関するオプションは、iostat のものと同じやり方で指定します。たとえば `mpstat -P 1 2 5` のように入力すると、2 秒間隔で 5 回、2 基目のプロセッサ (番号指定で 1) に関するレポートを出力します。

```
tux@mercury:~> mpstat -P 1 2 5
Linux 2.6.32.7-0.2-default (geeko@buildhost) 02/24/10 _x86_64_

08:57:10 CPU      %usr   %nice    %sys %iowait    %irq   %soft  %steal   %
%guest  %idle
08:57:12   1    4.46    0.00    5.94    0.50    0.00    0.00    0.00   %
0.00   89.11
08:57:14   1    1.98    0.00    2.97    0.99    0.00    0.99    0.00   %
0.00   93.07
08:57:16   1    2.50    0.00    3.00    0.00    0.00    1.00    0.00   %
0.00   93.50
08:57:18   1   14.36    0.00    1.98    0.00    0.00    0.50    0.00   %
0.00   83.17
08:57:20   1    2.51    0.00    4.02    0.00    0.00    2.01    0.00   %
0.00   91.46
Average:    1    5.17    0.00    3.58    0.30    0.00    0.90    0.00   %
```


2.2.3 タスクの監視: pidstat

お使いのシステムにおいて特定のタスクの負荷を確認したい場合は、pidstat コマンドを使用します。選択したそれぞれのタスクについて処理状況を表示するか、タスクを何も指定しない場合は Linux カーネルが管理する全タスクの処理状況を表示します。レポート数を指定することができるほか、それらの時間間隔を指定することもできます。

たとえば pidstat -C top 2 3 のように入力すると、コマンド名に「top」を含むタスクの負荷状況を表示します。2 秒間隔で計 3 回のレポートが作成されます。

```
tux@mercury:~> pidstat -C top 2 3
```

```
Linux 2.6.27.19-5-default (geeko@buildhost) 03/23/2009 _x86_64_
```

09:25:42 AM	PID	%usr	%system	%guest	%CPU	CPU	Command
09:25:44 AM	23576	37.62	61.39	0.00	99.01	1	top
09:25:44 AM	PID	%usr	%system	%guest	%CPU	CPU	Command
09:25:46 AM	23576	37.00	62.00	0.00	99.00	1	top
09:25:46 AM	PID	%usr	%system	%guest	%CPU	CPU	Command
09:25:48 AM	23576	38.00	61.00	0.00	99.00	1	top
Average:	PID	%usr	%system	%guest	%CPU	CPU	Command
Average:	23576	37.54	61.46	0.00	99.00	-	top

2.2.4 カーネルのリングバッファ表示: dmesg

Linux カーネルでは、特定のメッセージをリングバッファ内に保存しています。これらのメッセージを表示するには、dmesg コマンドをご利用ください:

```
tux@mercury:~> dmesg
```

```
[...]
```

```
end_request: I/O error, dev fd0, sector 0
subs: unsuccessful attempt to mount media (256)
e100: eth0: e100_watchdog: link up, 100Mbps, half-duplex
NET: Registered protocol family 17
IA-32 Microcode Update Driver: v1.14 <tigran@veritas.com>
microcode: CPU0 updated from revision 0xe to 0x2e, date = 08112004
IA-32 Microcode Update Driver v1.14 unregistered
boot splash: status on console 0 changed to on
NET: Registered protocol family 10
Disabled Privacy Extensions on device c0326ea0(lo)
IPv6 over IPv4 tunneling driver
powernow: This module only works with AMD K7 CPUs
boot splash: status on console 0 changed to on
```

さらに古いイベントについては、`/var/log/messages` や `/var/log/warn` ファイルに保存されます。

2.2.5 開かれているファイルの一覧表示: lsof

指定したプロセス ID (*PID*) で開いている 全ファイルを一覧表示するには、`-p` オプションを 指定します。たとえば、現在のシェルが開いている全てのファイルを確認するには、下記のように入力します:

```
tux@mercury:~$ lsof -p $$
COMMAND PID  USER  FD   TYPE DEVICE SIZE/OFF NODE NAME
bash    5552 tux    cwd   DIR   3, 3   1512 117619 /home/tux
bash    5552 tux    rtd   DIR   3, 3     584 2 /
bash    5552 tux    txt   REG   3, 3  498816 13047 /bin/bash
bash    5552 tux    mem   REG   0, 0      0 [heap] (stat: No such
bash    5552 tux    mem   REG   3, 3  217016 115687 /var/run/nscd/passwd
bash    5552 tux    mem   REG   3, 3  208464 11867 /usr/lib/locale/en_GB.
[...]
bash    5552 tux    mem   REG   3, 3     366 9720 /usr/lib/locale/en_GB.
bash    5552 tux    mem   REG   3, 3   97165 8828 /lib/ld-2.3.6.so
bash    5552 tux     0u   CHR  136, 5      7 /dev/pts/5
bash    5552 tux     1u   CHR  136, 5      7 /dev/pts/5
bash    5552 tux     2u   CHR  136, 5      7 /dev/pts/5
bash    5552 tux    255u  CHR  136, 5      7 /dev/pts/5
```

ここで `$$` は特別なシェル変数で、シェル自身の プロセス ID に置き換えられて実行されます。

また `lsof` に何もオプションを指定しないで実行すると、現在開かれている全てのファイルを一覧表示します。開かれている全ての ファイルを合計すると多数のファイルになってしまうため、全ファイルを表示してもあまり便利ではありません。その代わりに、検索機能を組み合わせることで、使いやすい一覧にするのがお勧めです。たとえばキャラクタデバイスの 一覧を表示するには、下記のように入力します:

```
tux@mercury:~$ lsof | grep CHR
bash    3838 tux     0u   CHR  136, 0      2 /dev/pts/0
bash    3838 tux     1u   CHR  136, 0      2 /dev/pts/0
bash    3838 tux     2u   CHR  136, 0      2 /dev/pts/0
bash    3838 tux    255u  CHR  136, 0      2 /dev/pts/0
bash    5552 tux     0u   CHR  136, 5      7 /dev/pts/5
bash    5552 tux     1u   CHR  136, 5      7 /dev/pts/5
bash    5552 tux     2u   CHR  136, 5      7 /dev/pts/5
bash    5552 tux    255u  CHR  136, 5      7 /dev/pts/5
X       5646 root    mem   CHR      1, 1    1006 /dev/mem
lsof    5673 tux     0u   CHR  136, 5      7 /dev/pts/5
lsof    5673 tux     2u   CHR  136, 5      7 /dev/pts/5
grep    5674 tux     1u   CHR  136, 5      7 /dev/pts/5
grep    5674 tux     2u   CHR  136, 5      7 /dev/pts/5
```

-i オプションを指定すると、lsof は インターネットファイルについても表示するようになります:

```
tux@mercury:~> lsof -i
[...]
pidgin      4349 tux    17r  IPv4  15194      0t0  TCP ¥
  jupiter.example.com:58542->www.example.net:https (ESTABLISHED)
pidgin      4349 tux    21u  IPv4  15583      0t0  TCP ¥
  jupiter.example.com:37051->aol.example.org:aol (ESTABLISHED)
evolution  4578 tux    38u  IPv4  16102      0t0  TCP ¥
  jupiter.example.com:57419->imap.example.com:imaps (ESTABLISHED)
npviewer.   9425 tux    40u  IPv4  24769      0t0  TCP ¥
  jupiter.example.com:51416->www.example.com:http (CLOSE_WAIT)
npviewer.   9425 tux    49u  IPv4  24814      0t0  TCP ¥
  jupiter.example.com:43964->www.example.org:http (CLOSE_WAIT)
ssh         17394 tux     3u   IPv4  40654      0t0  TCP ¥
  jupiter.example.com:35454->saturn.example.com:ssh (ESTABLISHED)
```

2.2.6 カーネルと udev のイベント順序監視: udevadm monitor

udevadm monitor コマンドは、udev のルールが発信した カーネルイベントと uevent を確認し、コンソールに対してそのイベントの デバイスパス (DEVPATH) を表示します。下記は USB メモリストICKを接続した場合のイベント例です:

注記: udev イベントの監視

root ユーザだけが udevadm コマンドを利用して udev イベントを監視することができます。

```
UEVENT[1138806687] add@/devices/pci0000:00/0000:00:1d.7/usb4/4-2/4-2.2
UEVENT[1138806687] add@/devices/pci0000:00/0000:00:1d.7/usb4/4-2/4-2.2/4-2.2
UEVENT[1138806687] add@/class/scsi_host/host4
UEVENT[1138806687] add@/class/usb_device/usbdev4.10
UDEV [1138806687] add@/devices/pci0000:00/0000:00:1d.7/usb4/4-2/4-2.2
UDEV [1138806687] add@/devices/pci0000:00/0000:00:1d.7/usb4/4-2/4-2.2/4-2.2
UDEV [1138806687] add@/class/scsi_host/host4
UDEV [1138806687] add@/class/usb_device/usbdev4.10
UEVENT[1138806692] add@/devices/pci0000:00/0000:00:1d.7/usb4/4-2/4-2.2/4-2.2
UEVENT[1138806692] add@/block/sdb
UEVENT[1138806692] add@/class/scsi_generic/sg1
UEVENT[1138806692] add@/class/scsi_device/4:0:0:0
UDEV [1138806693] add@/devices/pci0000:00/0000:00:1d.7/usb4/4-2/4-2.2/4-2.2
UDEV [1138806693] add@/class/scsi_generic/sg1
UDEV [1138806693] add@/class/scsi_device/4:0:0:0
UDEV [1138806693] add@/block/sdb
UEVENT[1138806694] add@/block/sdb/sdb1
UDEV [1138806694] add@/block/sdb/sdb1
```

UEVENT[1138806694] mount@/block/sdb/sdb1
UEVENT[1138806697] umount@/block/sdb/sdb1

2.2.7 セキュリティのイベント情報: audit

Linux の監査フレームワークは複雑な監査システムで、全てのセキュリティ関連イベントを収集します。これらの記録はその後セキュリティポリシーの違反が発生していないかどうかを調査するために分析が行なわれます。

2.2.8 X11 クライアントが利用しているサーバ資源情報の表示: xrestop

xrestop は、それぞれの接続済み X11 クライアントについて、サーバ側で消費されている資源情報を表示します。出力は 2.3.4 項「プロセス表: top」(25 ページ) とよく似た形になっています。

```
xrestop - Display: localhost:0
Monitoring 40 clients. XErrors: 0
Pixmaps: 42013K total, Other: 206K total, All: 42219K total
```

res-base	Wins	GCs	Fnts	Pxms	Misc	Pxm mem	Other	Total	PID	Identifier
3e00000	385	36	1	751	107	18161K	13K	18175K	?	NOVELL: SU
4600000	391	122	1	1182	889	4566K	33K	4600K	?	amaroK - S
1600000	35	11	0	76	142	3811K	4K	3816K	?	KDE Deskto
3400000	52	31	1	69	74	2816K	4K	2820K	?	Linux Shel
2c00000	50	25	1	43	50	2374K	3K	2378K	?	Linux Shel
2e00000	50	10	1	36	42	2341K	3K	2344K	?	Linux Shel
2600000	37	24	1	34	50	1772K	3K	1775K	?	Root - Kon
4800000	37	24	1	34	49	1772K	3K	1775K	?	Root - Kon
2a00000	209	33	1	323	238	1111K	12K	1123K	?	Trekstor25
1800000	182	32	1	302	285	1039K	12K	1052K	?	kicker
1400000	157	121	1	231	477	777K	18K	796K	?	kwin
3c00000	175	36	1	248	168	510K	9K	520K	?	de.comp.la
3a00000	326	42	1	579	444	486K	20K	506K	?	[opensuse-
0a00000	85	38	1	317	224	102K	9K	111K	?	Kopete
4e00000	25	17	1	60	66	63K	3K	66K	?	YaST Contr
2400000	11	10	0	56	51	53K	1K	55K	22061	suseplugge
0e00000	20	12	1	50	92	50K	3K	54K	22016	kded
3200000	6	41	5	72	84	40K	8K	48K	?	EMACS
2200000	54	9	1	30	31	42K	3K	45K	?	SUSEWatche
4400000	2	11	1	30	34	34K	2K	36K	16489	kdesu
1a00000	255	7	0	42	11	19K	6K	26K	?	KMix
3800000	2	14	1	34	37	21K	2K	24K	22242	knotify
1e00000	10	7	0	42	9	15K	624B	15K	?	KPowersave
3600000	106	6	1	30	9	7K	3K	11K	22236	konqueror
2000000	10	5	0	21	34	9K	1K	10K	?	klipper

2.3 プロセス

2.3.1 プロセス間通信: ipcs

ipcs プログラムは、現在使用中の IPC (プロセス間通信) を表示することができます:

```
----- Shared Memory Segments -----
key          shmid      owner      perms      bytes      nattch     status
0x00000000  58261504    tux        600        393216     2          dest
0x00000000  58294273    tux        600        196608     2          dest
0x00000000  83886083    tux        666        43264      2
0x00000000  83951622    tux        666        192000     2
0x00000000  83984391    tux        666        282464     2
0x00000000  84738056    root       644        151552     2          dest

----- Semaphore Arrays -----
key          semid      owner      perms      nsems
0x4d038abf   0          tux        600        8

----- Message Queues -----
key          msqid      owner      perms      used-bytes   messages
```

2.3.2 プロセスの一覧: ps

ps コマンドは、プロセスの一覧を表示することができます。ほとんどのパラメータはマイナス記号なしで書くことができます。簡潔な ヘルプを読む場合は `ps --help` と入力し、より詳しいヘルプを読む場合はマニュアルページをお読みください。

また、全てのプロセスについてユーザとコマンドライン情報を表示するには、`ps aux` と入力します:

```
tux@mercury:~> ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   696   272 ?        S    12:59   0:01 init [5]
root         2  0.0  0.0     0     0 ?        SN   12:59   0:00 [ksoftirqd
root         3  0.0  0.0     0     0 ?        S<   12:59   0:00 [events
[...]
tux      4047  0.0  6.0 158548 31400 ?        Ssl  13:02   0:06 mono-best
tux      4057  0.0  0.7   9036  3684 ?        Sl   13:02   0:00 /opt/gnome
tux      4067  0.0  0.1   2204   636 ?        S    13:02   0:00 /opt/gnome
tux      4072  0.0  1.0  15996  5160 ?        Ss   13:02   0:00 gnome-scre
```

```
tux 4114 0.0 3.7 130988 19172 ? SLl 13:06 0:04 sound-juic
tux 4818 0.0 0.3 4192 1812 pts/0 Ss 15:59 0:00 -bash
tux 4959 0.0 0.1 2324 816 pts/0 R+ 16:17 0:00 ps aux
```

たとえば sshd プロセスがいくつ動作しているのかを調べるには、`-p` オプションと `pidof` コマンドを併用してください。`pidof` コマンドは、指定された名前のコマンドを実行しているプロセス ID を、一覧表示するコマンドです。

```
tux@mercury:~> ps -p $(pidof sshd)
  PID TTY          STAT       TIME COMMAND
 3524 ?            Ss        0:00 /usr/sbin/sshd -o PidFile=/var/run/sshd.init.pid
 4813 ?            Ss        0:00 sshd: tux [priv]
 4817 ?            R         0:00 sshd: tux@pts/0
```

プロセス一覧の書式は必要に応じて変更することができます。`-L` オプションを指定して実行すると、書式に関する全てのキーワードを出力することができます。たとえば下記のコマンドでは、全てのプロセスをメモリの使用率順に並べて表示します：

```
tux@mercury:~> ps ax --format pid,rss,cmd --sort rss
  PID  RSS CMD
    2     0 [ksoftirqd/0]
    3     0 [events/0]
    4     0 [khelper]
    5     0 [kthread]
   11     0 [kblockd/0]
   12     0 [kacpid]
  472     0 [pdflush]
  473     0 [pdflush]
[...]
4028 17556 nautilus --no-default-window --sm-client-id default2
4118 17800 ksnapshot
4114 19172 sound-juicer
4023 25144 gnome-panel --sm-client-id default1
```

便利な `ps` 呼び出し

```
ps aux --sort 列
```

列の出力順に並べ替えます。それぞれ 列 には下記を指定できます。

物理メモリの比率: `pmem`

CPU の使用率: `pcpu`

常駐セットサイズ (スワップしていない物理メモリ): `rss`

```
ps axo pid,%cpu,rss,vsz,args,wchan
```

各プロセスについて PID と CPU の使用率、メモリサイズ (常駐および仮想) と名前、およびシステムコール名を表示します。

```
ps axfo pid,args
```

プロセスツリー (木構造) を表示します。

2.3.3 プロセスのツリー表示: pstree

pstree コマンドは、プロセスの一覧をツリー (木) 構造 で表示します:

```
tux@mercury:~> pstree
init--+-NetworkManagerD
      |-acpid
      |-3*[automount]
      |-cron
      |-cupsd
      |-2*[dbus-daemon]
      |-dbus-launch
      |-dcopserver
      |-dhcpcd
      |-events/0
      |-gpg-agent
      |-hald--+-hald-addon-acpi
              `--hald-addon-stor
      |-kded
      |-kdeinit--+-kdesu---su---kdesu_stub---yast2---y2controlcenter
                  |
                  |   |--kio_file
                  |   |--klauncher
                  |   |--konqueror
                  |   |--konsole--+-bash---su---bash
                  |               `--bash
                  |
                  |   `--kwin
      |-kdesktop---kdesktop_lock---xmatrix
      |-kdesud
      |-kdm--+-X
              `--kdm---startkde---kwrapper
[...]
```

-p オプションを指定すると、それぞれの名前の後ろに プロセス ID を表示することができます。コマンドラインについても表示する には、-a を指定してください。

2.3.4 プロセス表: top

top コマンドは "table of processes" (プロセス表) の 略で、2 秒おきに更新するタイプのプロセス一覧を表示します。プログラムを 終了するには、Q を押してください。また、-n 1 パラメータを指定すると、プロセス一覧を一度表示 するだけでプログラムが終了するようになります。下記は、top -n 1 の出力例です:

```
tux@mercury:~> top -n 1
top - 17:06:28 up 2:10, 5 users, load average: 0.00, 0.00, 0.00
Tasks: 85 total, 1 running, 83 sleeping, 1 stopped, 0 zombie
Cpu(s): 5.5% us, 0.8% sy, 0.8% ni, 91.9% id, 1.0% wa, 0.0% hi, 0.0% si
Mem: 515584k total, 506468k used, 9116k free, 66324k buffers
Swap: 658656k total, 0k used, 658656k free, 353328k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
-----	------	----	----	------	-----	-----	---	------	------	-------	---------

```

1 root      16   0   700 272 236 S  0.0  0.1  0:01.33 init
2 root      34  19    0   0   0 S  0.0  0.0  0:00.00 ksoftirqd/0
3 root      10  -5    0   0   0 S  0.0  0.0  0:00.27 events/0
4 root      10  -5    0   0   0 S  0.0  0.0  0:00.01 khelper
5 root      10  -5    0   0   0 S  0.0  0.0  0:00.00 kthread
11 root     10  -5    0   0   0 S  0.0  0.0  0:00.05 kblockd/0
12 root     20  -5    0   0   0 S  0.0  0.0  0:00.00 kacpid
472 root     20   0    0   0   0 S  0.0  0.0  0:00.00 pdflush
473 root     15   0    0   0   0 S  0.0  0.0  0:00.06 pdflush
475 root     11  -5    0   0   0 S  0.0  0.0  0:00.00 aio/0
474 root     15   0    0   0   0 S  0.0  0.0  0:00.07 kswapd0
681 root     10  -5    0   0   0 S  0.0  0.0  0:00.01 kseriod
839 root     10  -5    0   0   0 S  0.0  0.0  0:00.02 reiserfs/0
923 root     13  -4  1712 552 344 S  0.0  0.1  0:00.67 udevd
1343 root    10  -5    0   0   0 S  0.0  0.0  0:00.00 khubd
1587 root    20   0    0   0   0 S  0.0  0.0  0:00.00 shpchpd_event
1746 root    15   0    0   0   0 S  0.0  0.0  0:00.00 wl_control
1752 root    15   0    0   0   0 S  0.0  0.0  0:00.00 wl_bus_master1
2151 root    16   0  1464 496 416 S  0.0  0.1  0:00.00 acpid
2165 messageb 16   0  3340 1048 792 S  0.0  0.2  0:00.64 dbus-daemon
2166 root    15   0  1840 752 556 S  0.0  0.1  0:00.01 syslog-ng
2171 root    16   0  1600 516 320 S  0.0  0.1  0:00.00 klogd
2235 root    15   0  1736 800 652 S  0.0  0.2  0:00.10 resmgrp
2289 root    16   0  4192 2852 1444 S  0.0  0.6  0:02.05 hald
2403 root    23   0  1756 600 524 S  0.0  0.1  0:00.00 hald-addon-acpi
2709 root    19   0  2668 1076 944 S  0.0  0.2  0:00.00 NetworkManagerD
2714 root    16   0  1756 648 564 S  0.0  0.1  0:00.56 hald-addon-stor

```

既定では CPU の使用率順 (%CPU の列、ショートカットでは + P) に並べられています。それぞれ下記のショートカットで並び順を変更することができます:

- + M: 常駐メモリ (RES)
- + N: プロセス ID (PID)
- + T: CPU 時間 (TIME+)

それ以外の項目を並べ替えに使用するには、F を押して一覧から 項目を選択します。並べ替え順序 (昇順、降順) を切り替えるには、+ R を押します。

また、-U *UID* オプションを 指定すると、特定のユーザのプロセスだけを監視するようになります。ここで、*UID* にはユーザのユーザ ID を指定してください。たとえば、top -U \$(id -u) のように入力すると、自分自身が起動したプロセスについて一覧表示を行ないます。

2.3.5 top コマンドに似た I/O モニタ: iotop

iotop ユーティリティは、プロセスやスレッドごとに I/O 使用率を表形式で表示することができるユーティリティです。

ヒント

iotop は既定ではインストールされません。root で `zypper in iotop` を実行し、インストールを行なってください。

iotop は指定された採取間隔の間に発生した I/O について、その読み込みと書き込み帯域をプロセスごとに列で表示します。また、各プロセス がスワップへの書き込み処理に費やした時間の割合や、I/O の完了を待機していた時間の割合を表示することもできます。それ以外にも、それぞれの I/O 優先順位 (分類／レベル) も表示されます。さらに、採取間隔の間に発生した 読み込みと書き込みの I/O 帯域の合計値も、インターフェイスの冒頭部に 表示します。

並び順を変えるには、それぞれ左右のカーソルキーを使用します。また R では並び順を逆順に、O は `--only` オプションの切り替えを、P は `--processes` オプションの切り替えをそれぞれ行ないます。さらに A は `--accumulated` オプションの 切り替えを、Q はプログラムの終了を行なうことができます。また、I はスレッドやプロセスの優先順位を変更することができます。それ以外のキーは全て更新を行なうためのキーです。

下記は `iotop --only` の出力例です。find と emacs が動作中である 状況が見えています:

```
tux@mercury:~$ iotop --only
Total DISK READ: 50.61 K/s | Total DISK WRITE: 11.68 K/s
  TID  PRIO  USER      DISK READ  DISK WRITE  SWAPIN     IO>     COMMAND
 3416  be/4  ke        50.61 K/s   0.00 B/s   0.00 %   4.05 %  find /
  275  be/3  root       0.00 B/s   3.89 K/s   0.00 %   2.34 %  [jbd2/sda2-8]
 5055  be/4  ke         0.00 B/s   3.89 K/s   0.00 %   0.04 %  emacs
```

iotop はバッチモード (-b) を 使用することができるほか、ファイルに出力して後から分析を行なうことも できます。オプションの一覧について、詳しくはマニュアルページ (man 1 iotop) をお読みください。

2.3.6 プロセスの nice 値の変更: nice と renice

カーネルは各プロセスの優先レベル (nice 値とも呼びます) をもとにして、どのプロセスに対してより多くの CPU 時間を与えるかを決定します。あるプロセスに対してより大きい「nice」値が設定されている 場合、プロセスに与えられる CPU 時間は少なくなります。nice 値は -20 (最優先) から 19 までの間で設定します。マイナスの値は root だけが設定できます。

nice 値のレベルを調整すると、処理時間が長くなるだけでなく CPU 時間も 大きく消費し、時間面の制約が少ないプログラム、たとえばカーネルの コンパイル作業を動作させるような場合に効果があります。それらのプロセスの「nice 値を大きくする」ことで、Web サーバなどの他の処理を 優先的に動作させることができます。

また、オプションを何も指定せずに nice を実行すると、現在の nice 値を表示することができます：

```
tux@mercury:~> nice
0
```

nice コマンドのように 実行すると、指定したコマンドの nice 値を 10 だけプラスします。また、nice -n レベル コマンドのように実行すると、現在の nice 値に対する相対値で指定することができます。

実行中のプロセスに対する nice 値を変更するには、renice 優先度 -p プロセス id のように実行します。たとえば：

```
renice +5 3266
```

指定したユーザの全プロセスに対する nice 値を変更するには、-u ユーザ オプションをご利用ください。また、プロセスグループを指定する場合は、-g プロセスグループ ID のように指定します。

2.4 メモリ

2.4.1 メモリの使用量: free

free ユーティリティは、メモリの使用率を表示することができます。空き容量の詳細や使用済み容量の詳細、スワップ領域の詳細がそれぞれ表示されます：

```
tux@mercury:~> free
```

	total	used	free	shared	buffers	cached
Mem:	2062844	2047444	15400	0	129580	921936
-/+ buffers/cache:		995928	1066916			
Swap:	2104472	0	2104472			

それぞれ -b, -k, -m, -g オプションを指定すると、バイト単位／キロバイト単位／メガバイト単位／ギガバイト単位に値を表示ようになります。また、-d 時間 オプションを指定すると、時間 で指定した秒間隔ごとに表示が更新されるようになります。たとえば free -d 1.5 と入力すると、1.5 秒間隔で更新が行なわれます。

2.4.2 詳細なメモリ使用量: /proc/meminfo

/proc/meminfo を利用することで、free が表示するデータよりもさらに詳細な情報を得ることができます。実際のところ、free はここからの出力を利用しています。下記に 64 ビット システムでの表示例を示します。32 ビットシステムの場合、メモリの管理形態が異なる都合上、出力項目が若干異なることに注意してください:

```
tux@mercury:~> cat /proc/meminfo
MemTotal:      8182956 kB
MemFree:       1045744 kB
Buffers:       364364 kB
Cached:        5601388 kB
SwapCached:    1936 kB
Active:        4048268 kB
Inactive:      2674796 kB
Active(anon):  663088 kB
Inactive(anon): 107108 kB
Active(file):  3385180 kB
Inactive(file): 2567688 kB
Unevictable:   4 kB
Mlocked:       4 kB
SwapTotal:     2096440 kB
SwapFree:      2076692 kB
Dirty:         44 kB
Writeback:     0 kB
AnonPages:     756108 kB
Mapped:        147320 kB
Slab:          329216 kB
SReclaimable:  300220 kB
SUnreclaim:    28996 kB
PageTables:    21092 kB
NFS_Unstable:  0 kB
Bounce:        0 kB
WritebackTmp:  0 kB
CommitLimit:   6187916 kB
Committed_AS:  1388160 kB
VmallocTotal:  34359738367 kB
VmallocUsed:    133384 kB
VmallocChunk:  34359570939 kB
HugePages_Total: 0
HugePages_Free: 0
HugePages_Rsvd: 0
HugePages_Surp: 0
Hugepagesize:  2048 kB
DirectMap4k:   2689024 kB
DirectMap2M:   5691392 kB
```

最も重要な項目は下記のとおりです:

MemTotal

利用可能な RAM の合計値

MemFree

未使用の RAM の合計値

Buffers

RAM 内のファイルバッファキャッシュ

Cached

RAM 内のページキャッシュ (バッファキャッシュを除く)

SwapCached

スワップ内のページキャッシュ

Active

通常は再利用のできない、直近で使用されたメモリ。この値は匿名ページ (*Active(anon)*) として一覧表示されるもの) によって 要求されたメモリと、ファイルバックのページ (*Active(file)*) の合計値を示しています。

Inactive

再利用の可能な、直近では使用されていないメモリ。この値は匿名ページ (*Inactive(anon)*) として一覧表示されるもの) によって 要求されたメモリと、ファイルバックページ (*Inactive(file)*) の合計値を示しています。

SwapTotal

スワップ領域の合計サイズ

SwapFree

未使用のスワップ領域の合計サイズ

Dirty

ディスクに書き込まれる予定のメモリ量

Writeback

現在ディスクに書き込んでいる最中のメモリ量

Mapped

nmap コマンドで割り当てられたメモリ

Slab

カーネルのデータ構造キャッシュ

SReclaimable

再利用可能な SLAB キャッシュ (inode, dentry など)

Committed_AS

現在の負荷量を処理するにあたって必要なメモリ量 (RAM とスワップ) の 合計
値の見積もり

2.4.3 プロセスのメモリ使用量: smaps

特定のプロセスがどれだけメモリを利用しているのかを正確に知るには、top や ps のような標準ツールでは 判断できません。このような要件には、カーネル 2.6.14 から提供されるようになった smaps サブシステムをお使いください。smaps サブシステムは /proc/プロセス ID/smaps からアクセスできます。なお プロセス ID は実際のプロセス ID に置き換えてください。このサブシステムは共有メモリと プライベートメモリを区別して表示するため、他のプロセスと共有されておらず 独占的に使用されているメモリ量を判断することができます。

2.5 ネットワーク

2.5.1 ネットワーク状態の表示: netstat

netstat はネットワーク接続を表示することができるほか、ルーティングテーブル (-r), インターフェイス (-i), マスカレード接続 (-M), マルチキャストのメンバーシップ情報 (-g), 統計情報 (-s) をそれぞれ表示することができます。

```
tux@mercury:~> netstat -r
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
192.168.2.0	*	255.255.254.0	U	0	0	0	eth0
link-local	*	255.255.0.0	U	0	0	0	eth0
loopback	*	255.0.0.0	U	0	0	0	lo
default	192.168.2.254	0.0.0.0	UG	0	0	0	eth0

```
tux@mercury:~> netstat -i
```

Kernel Interface table

Iface	MTU	Met	RX-OK	RX-ERR	RX-DRP	RX-OVR	TX-OK	TX-ERR	TX-DRP	TX-OVR	Flg
eth0	1500	0	1624507	129056	0	0	7055	0	0	0	BMNRU
lo	16436	0	23728	0	0	0	23728	0	0	0	LRU

ネットワーク接続や統計情報を表示する際、表示するソケットタイプを指定することができます。それぞれ TCP (-t), UDP (-u), raw (-r) を指定してください。また、-p を指定すると、各ソケットの属している PID とプログラム名を表示します。

下記の例では、全ての TCP 接続とそれらの接続を使用しているプログラムを表示しています:

```
mercury:~ # netstat -t -p
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Pro
[...]
tcp      0      0 mercury:33513 www.novell.com:www-http ESTABLISHED 6862/fi
tcp      0    352 mercury:ssh mercury2.:trc-netpoll ESTABLISHED 19422/s
tcp      0      0 localhost:ssh localhost:17828 ESTABLISHED -
```

下記の例では、TCP プロトコルの統計情報を表示しています:

```
tux@mercury:~> netstat -s -t
Tcp:
  2427 active connections openings
  2374 passive connection openings
   0 failed connection attempts
   0 connection resets received
   1 connections established
 27476 segments received
 26786 segments send out
   54 segments retransmitted
   0 bad segments received.
   6 resets sent
[...]
TCPAbortOnLinger: 0
TCPAbortFailed: 0
TCPMemoryPressures: 0
```

2.5.2 対話的なネットワークモニタ: iptraf

iptraf はメニューベースのローカルエリアネットワーク (LAN) モニタです。TCP, UDP のカウントやイーサネットの負荷情報、IP チェックサムエラーなどのネットワーク統計情報を生成します。

ヒント

iptraf は既定ではインストールされません。インストール を行なうには、root で `zypper in iptraf` を実行 してください。

何もオプションを指定せずに実行した場合は、対話モードで動作します。グラフィカルなメニューを利用することができるほか、iptraf で収集したい統計情報を選択することができます。ネットワークインターフェイスを 指定することもできます。

図 2.1 iptraf の対話モード実行

	Total Packets	Total Bytes	Incoming Packets	Incoming Bytes	Outgoing Packets	Outgoing Bytes
Total:	1577	984734	881	906268	696	78466
IP:	1577	961844	881	893122	696	68722
TCP:	1466	948809	827	883927	639	64882
UDP:	110	12815	54	9195	56	3620
ICMP:	1	220	0	0	1	220
Other IP:	0	0	0	0	0	0
Non-IP:	0	0	0	0	0	0

Total rates:	706.7 kbits/sec	Broadcast packets:	0
	113.8 packets/sec	Broadcast bytes:	0
Incoming rates:	677.3 kbits/sec		
	69.0 packets/sec		
Outgoing rates:	29.4 kbits/sec	IP checksum errors:	0
	44.8 packets/sec		

Elapsed time: 0:00
X=exit

また、iptraf コマンドには複数のオプションがあり、バッチモードでも動作させることができます。下記の例では、ネットワーク インターフェイス eth0 (-i) の統計情報を 1 分間収集する 動作をします。さらに、本プログラムはバックグラウンド (-B) で動作させることもできます。この場合、統計情報はお使いのホームディレクトリ 内にある iptraf.log ファイルに出力されます (-L)。

```
tux@mercury:~> iptraf -i eth0 -t 1 -B -L ~/iptraf.log
```

生成されたログファイルは、more コマンドで閲覧できます:

```
tux@mercury:~> more ~/iptraf.log
Mon Mar 23 10:08:02 2010; ***** IP traffic monitor started *****
Mon Mar 23 10:08:02 2010; UDP; eth0; 107 bytes; from 192.168.1.192:33157 to ¥
239.255.255.253:427
Mon Mar 23 10:08:02 2010; VRRP; eth0; 46 bytes; from 192.168.1.252 to ¥
224.0.0.18
Mon Mar 23 10:08:03 2010; VRRP; eth0; 46 bytes; from 192.168.1.252 to ¥
224.0.0.18
Mon Mar 23 10:08:03 2010; VRRP; eth0; 46 bytes; from 192.168.1.252 to ¥
224.0.0.18
[...]
Mon Mar 23 10:08:06 2010; UDP; eth0; 132 bytes; from 192.168.1.54:54395 to ¥
10.20.7.255:111
Mon Mar 23 10:08:06 2010; UDP; eth0; 46 bytes; from 192.168.1.92:27258 to ¥
10.20.7.255:8765
Mon Mar 23 10:08:06 2010; UDP; eth0; 124 bytes; from 192.168.1.139:43464 to ¥
10.20.7.255:111
Mon Mar 23 10:08:06 2010; VRRP; eth0; 46 bytes; from 192.168.1.252 to ¥
224.0.0.18
--More--(7%)
```

2.6 /proc ファイルシステム

/proc ファイルシステムはカーネルが提供する擬似的な ファイルシステムで、ファイルの形式で各種の重要情報を提供します。たとえば下記のコマンドを入力すると、CPU の種類を表示することができます:

```
tux@mercury:~> cat /proc/cpuinfo
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 15
model          : 4
model name     : Intel(R) Pentium(R) 4 CPU 3.40GHz
stepping       : 3
cpu MHz        : 2800.000
cache size     : 2048 KB
physical id    : 0
[...]
```

また、割り込みの割り当てや使用について調べるには、下記のようなコマンドを入力します:

```
tux@mercury:~> cat /proc/interrupts
CPU0
0:   3577519      XT-PIC  timer
1:      130      XT-PIC  i8042
2:         0      XT-PIC  cascade
5:   564535      XT-PIC  Intel 82801DB-ICH4
7:         1      XT-PIC  parport0
8:         2      XT-PIC  rtc
9:         1      XT-PIC  acpi, uhci_hcd:usb1, ehci_hcd:usb4
10:        0      XT-PIC  uhci_hcd:usb3
11:   71772      XT-PIC  uhci_hcd:usb2, eth0
12:   101150      XT-PIC  i8042
14:   33146      XT-PIC  ide0
15:  149202      XT-PIC  ide1
NMI:         0
LOC:         0
ERR:         0
MIS:         0
```

下記に /proc 以下にある重要なファイルと、その内容説明を記します:

/proc/devices

利用可能なデバイスの一覧

/proc/modules

読み込まれたカーネルモジュール

/proc/cmdline

カーネルのコマンドラインパラメータ

/proc/meminfo

メモリの使用率に関する詳細情報

/proc/config.gz

現在実行中のカーネルについて、そのカーネル設定ファイルを gzip 圧縮したもの

さらなる情報は、/usr/src/linux/Documentation/filesystems/proc.txt のテキストファイル内に書かれています (なお、このファイルは kernel-source パッケージをインストールした場合にのみ利用できます)。また、現在実行中のプロセスに関する情報は、/proc/*NNN* ディレクトリ 以下から読み取ることができます。ここで *NNN* にはプロセス ID (PID) を指定します。自分自身のプロセスについて調べる場合は、/proc/self/ を利用して調べることができます:

```
tux@mercury:~> ls -l /proc/self
lrwxrwxrwx 1 root root 64 2007-07-16 13:03 /proc/self -> 5356
tux@mercury:~> ls -l /proc/self/
total 0
dr-xr-xr-x 2 tux users 0 2007-07-16 17:04 attr
-r----- 1 tux users 0 2007-07-16 17:04 auxv
-r--r--r-- 1 tux users 0 2007-07-16 17:04 cmdline
lrwxrwxrwx 1 tux users 0 2007-07-16 17:04 cwd -> /home/tux
-r----- 1 tux users 0 2007-07-16 17:04 environ
lrwxrwxrwx 1 tux users 0 2007-07-16 17:04 exe -> /bin/ls
dr-x----- 2 tux users 0 2007-07-16 17:04 fd
-rw-r--r-- 1 tux users 0 2007-07-16 17:04 loginuid
-r--r--r-- 1 tux users 0 2007-07-16 17:04 maps
-rw----- 1 tux users 0 2007-07-16 17:04 mem
-r--r--r-- 1 tux users 0 2007-07-16 17:04 mounts
-rw-r--r-- 1 tux users 0 2007-07-16 17:04 oom_adj
-r--r--r-- 1 tux users 0 2007-07-16 17:04 oom_score
lrwxrwxrwx 1 tux users 0 2007-07-16 17:04 root -> /
-rw----- 1 tux users 0 2007-07-16 17:04 seccomp
-r--r--r-- 1 tux users 0 2007-07-16 17:04 smaps
-r--r--r-- 1 tux users 0 2007-07-16 17:04 stat
[...]
dr-xr-xr-x 3 tux users 0 2007-07-16 17:04 task
-r--r--r-- 1 tux users 0 2007-07-16 17:04 wchan
```

実行ファイルとライブラリのアドレス割り当ては、maps ファイル内に書かれています:

```
tux@mercury:~> cat /proc/self/maps
08048000-0804c000 r-xp 00000000 03:03 17753      /bin/cat
0804c000-0804d000 rw-p 00004000 03:03 17753      /bin/cat
0804d000-0806e000 rw-p 0804d000 00:00 0         [heap]
b7d27000-b7d5a000 r--p 00000000 03:03 11867      /usr/lib/locale/en_GB.utf8/
b7d5a000-b7e32000 r--p 00000000 03:03 11868      /usr/lib/locale/en_GB.utf8/
b7e32000-b7e33000 rw-p b7e32000 00:00 0
b7e33000-b7f45000 r-xp 00000000 03:03 8837       /lib/libc-2.3.6.so
b7f45000-b7f46000 r--p 00112000 03:03 8837       /lib/libc-2.3.6.so
b7f46000-b7f48000 rw-p 00113000 03:03 8837       /lib/libc-2.3.6.so
```

```

b7f48000-b7f4c000 rw-p b7f48000 00:00 0
b7f52000-b7f53000 r--p 00000000 03:03 11842 /usr/lib/locale/en_GB.utf8/
[...]
b7f5b000-b7f61000 r--s 00000000 03:03 9109 /usr/lib/gconv/gconv-module
b7f61000-b7f62000 r--p 00000000 03:03 9720 /usr/lib/locale/en_GB.utf8/
b7f62000-b7f66000 r-xp 00000000 03:03 8828 /lib/ld-2.3.6.so
b7f66000-b7f68000 rw-p 00013000 03:03 8828 /lib/ld-2.3.6.so
bfd61000-bfd67000 rw-p bfd61000 00:00 0 [stack]
ffffe000-fffff000 ---p 00000000 00:00 0 [vdso]

```

2.6.1 procinfo

/proc ファイルシステムにある重要な情報は、procinfo コマンドを利用することで概要表示を行なうことができます:

```

tux@mercury:~> procinfo
Linux 2.6.32.7-0.2-default (geeko@buildhost) (gcc 4.3.4) #1 2CPU

Memory:      Total      Used      Free      Shared    Buffers
Mem:         2060604    2011264    49340      0        200664
Swap:        2104472      112     2104360

Bootup: Wed Feb 17 03:39:33 2010    Load average: 0.86 1.10 1.11 3/118 21547

user  :      2:43:13.78   0.8% page in :   71099181 disk 1: 2827023r 968
nice  :    1d 22:21:27.87  14.7% page out:  690734737
system:   13:39:57.57    4.3% page act:  138388345
IOWait:   18:02:18.59    5.7% page dea:  29639529
hw irq:    0:03:39.44    0.0% page flt: 9539791626
sw irq:    1:15:35.25    0.4% swap in :      69
idle  :    9d 16:07:56.79 73.8% swap out:    209
uptime:   6d 13:07:11.14 context :  542720687

irq 0: 141399308 timer      irq 14: 5074312 ide0
irq 1:   73784 i8042        irq 50: 1938076 uhci_hcd:usb1, ehci_
irq 4:      2             irq 58:      0 uhci_hcd:usb2
irq 6:      5 floppy [2]   irq 66:  872711 uhci_hcd:usb3, HDA I
irq 7:      2             irq 74:      15 uhci_hcd:usb4
irq 8:      0 rtc          irq 82: 178717720 0 PCI-MSI e
irq 9:      0 acpi         irq169: 44352794 nvidia
irq 12:     3             irq233:  8209068 0 PCI-MSI l

```

全ての情報を表示するには、-a オプションを指定してください。-nN オプションを指定すると、N 秒おきに情報を更新するようになります。この場合は、q を押すとプログラムを終了することができます。

既定では累積値を表示するようになっていますが、-d を指定 すると差を表示するようになります。たとえば procinfo -dn5 と入力すると、最近 5 秒間に変更のあった値が表示されます。

2.7 ハードウェア情報

2.7.1 PCI 資源: lspci

注記: PCI 設定へのアクセスについて

コンピュータの PCI 設定にアクセスするには、多くのオペレーティングシステムで root のユーザ権限が必要です。

lspci コマンドは PCI 資源の一覧を表示します:

```
mercury:~ # lspci
00:00.0 Host bridge: Intel Corporation 82845G/GL[Brookdale-G]/GE/PE ¥
    DRAM Controller/Host-Hub Interface (rev 01)
00:01.0 PCI bridge: Intel Corporation 82845G/GL[Brookdale-G]/GE/PE ¥
    Host-to-AGP Bridge (rev 01)
00:1d.0 USB Controller: Intel Corporation 82801DB/DBL/DBM ¥
    (ICH4/ICH4-L/ICH4-M) USB UHCI Controller #1 (rev 01)
00:1d.1 USB Controller: Intel Corporation 82801DB/DBL/DBM ¥
    (ICH4/ICH4-L/ICH4-M) USB UHCI Controller #2 (rev 01)
00:1d.2 USB Controller: Intel Corporation 82801DB/DBL/DBM ¥
    (ICH4/ICH4-L/ICH4-M) USB UHCI Controller #3 (rev 01)
00:1d.7 USB Controller: Intel Corporation 82801DB/DBM ¥
    (ICH4/ICH4-M) USB2 EHCI Controller (rev 01)
00:1e.0 PCI bridge: Intel Corporation 82801 PCI Bridge (rev 81)
00:1f.0 ISA bridge: Intel Corporation 82801DB/DBL (ICH4/ICH4-L) ¥
    LPC Interface Bridge (rev 01)
00:1f.1 IDE interface: Intel Corporation 82801DB (ICH4) IDE ¥
    Controller (rev 01)
00:1f.3 SMBus: Intel Corporation 82801DB/DBL/DBM (ICH4/ICH4-L/ICH4-M) ¥
    SMBus Controller (rev 01)
00:1f.5 Multimedia audio controller: Intel Corporation 82801DB/DBL/DBM ¥
    (ICH4/ICH4-L/ICH4-M) AC'97 Audio Controller (rev 01)
01:00.0 VGA compatible controller: Matrox Graphics, Inc. G400/G450 (rev 85)
02:08.0 Ethernet controller: Intel Corporation 82801DB PRO/100 VE (LOM) ¥
    Ethernet Controller (rev 81)
```

-v オプションを使用すると、さらに詳細な一覧を表示することができます:

```
mercury:~ # lspci -v
[...]
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet ¥
    Controller (rev 02)
    Subsystem: Intel Corporation PRO/1000 MT Desktop Adapter
    Flags: bus master, 66MHz, medium devsel, latency 64, IRQ 19
    Memory at f0000000 (32-bit, non-prefetchable) [size=128K]
```

```
I/O ports at d010 [size=8]
Capabilities: [dc] Power Management version 2
Capabilities: [e4] PCI-X non-bridge device
Kernel driver in use: e1000
Kernel modules: e1000
```

デバイス名の解決は /usr/share/pci.ids ファイルを利用して行なわれます。このファイル内に対応する PCI ID とその名前が存在しない場合、「Unknown device」(未知のデバイス)として表示されます。

-vv オプションを使用すると、プログラムで問い合わせ可能な全ての情報を出力します。また、名前ではなく数値で表示させたい場合は、-n オプションを使用してください。

2.7.2 USB デバイス: lsusb

lsusb コマンドは全ての USB デバイスを一覧表示します。-v オプションを指定すると、さらに詳しい一覧が表示されます。詳しい一覧は /proc/bus/usb/ ディレクトリから読み出す仕組みになっています。下記は USB ハブ、USB メモリ、ハードディスク、マウスがそれぞれ USB 経由で接続されている場合の例です。

```
mercury:/ # lsusb
Bus 004 Device 007: ID 0ea0:2168 Ours Technology, Inc. Transcend JetFlash ¥
2.0 / Astone USB Drive
Bus 004 Device 006: ID 04b4:6830 Cypress Semiconductor Corp. USB-2.0 IDE ¥
Adapter
Bus 004 Device 005: ID 05e3:0605 Genesys Logic, Inc.
Bus 004 Device 001: ID 0000:0000
Bus 003 Device 001: ID 0000:0000
Bus 002 Device 001: ID 0000:0000
Bus 001 Device 005: ID 046d:c012 Logitech, Inc. Optical Mouse
Bus 001 Device 001: ID 0000:0000
```

2.8 ファイルとファイルシステム

2.8.1 ファイル種類の判別: file

file コマンドは、指定した単一または複数のファイルについて、/usr/share/misc/magic から読み込んだ データを元に種類を判別します。

```
tux@mercury:~> file /usr/bin/file
/usr/bin/file: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), ¥
```

for GNU/Linux 2.6.4, dynamically linked (uses shared libs), stripped

`-f` 一覧を指定すると、種類を判別したいファイル名の一覧が書かれたファイルを指定することができます。また、`-z` を指定すると、`file` が圧縮ファイル内の判別を行なうようになります:

```
tux@mercury:~> file /usr/share/man/man1/file.1.gz
/usr/share/man/man1/file.1.gz: gzip compressed data, from Unix, max compression
tux@mercury:~> file -z /usr/share/man/man1/file.1.gz
/usr/share/man/man1/file.1.gz: troff or preprocessor input text ¥
(gzip compressed data, from Unix, max compression)
```

また、`-i` を指定すると、従来の表記法ではなく MIME タイプ の文字列を表示します。

```
tux@mercury:~> file -i /usr/share/misc/magic
/usr/share/misc/magic: text/plain charset=utf-8
```

2.8.2 ファイルシステムとその使用率: mount, df, du

`mount` コマンドは、各マウントポイントにどのファイル システムがマウントされているかを表示します:

```
tux@mercury:~> mount
/dev/sda2 on / type ext4 (rw,acl,user_xattr)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
debugfs on /sys/kernel/debug type debugfs (rw)
devtmpfs on /dev type devtmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,mode=1777)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
/dev/sda3 on /home type ext3 (rw)
securityfs on /sys/kernel/security type securityfs (rw)
fusectl on /sys/fs/fuse/connections type fusectl (rw)
gvfs-fuse-daemon on /home/tux/.gvfs type fuse.gvfs-fuse-daemon ¥
(rw,nosuid,nodev,user=tux)
```

ファイルシステム内での使用率に関する情報を得たい場合は、`df` をお使いください。`-h` (または `--human-readable`) オプションを指定すると、一般的なユーザにとって、より読みやすい形式で表示することができます。

```
tux@mercury:~> df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda2       20G   5,9G   13G   32% /
devtmpfs        1,6G   236K   1,6G    1% /dev
tmpfs           1,6G   668K   1,6G    1% /dev/shm
/dev/sda3       208G   40G   159G   20% /home
```

また、指定したディレクトリ内のファイルと、そのサブディレクトリ以下にあるファイルについてサイズを確認するには、`du` を使用します。`-s` オプションを指定すると、詳細な情報 出力を省いて、パラメータで指定した各ディレクトリの合計サイズだけを表示します。こちらも `-h` オプションを指定することができ、同じく一般的なユーザにとって読みやすい形式で表示することができます:

```
tux@mercury:~> du -sh /opt
192M    /opt
```

2.8.3 ELF バイナリに関する追加情報表示

バイナリファイルの内容を読みたい場合は、`readelf` をお使いください。このコマンドでは、他のアーキテクチャ向けに作成された ELF 形式のファイルを読み取ることもできます:

```
tux@mercury:~> readelf --file-header /bin/ls
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00
  Class:                                ELF64
  Data:                                   2's complement, little endian
  Version:                               1 (current)
  OS/ABI:                                 UNIX - System V
  ABI Version:                           0
  Type:                                   EXEC (Executable file)
  Machine:                               Advanced Micro Devices X86-64
  Version:                               0x1
  Entry point address:                   0x402540
  Start of program headers:              64 (bytes into file)
  Start of section headers:             95720 (bytes into file)
  Flags:                                  0x0
  Size of this header:                   64 (bytes)
  Size of program headers:               56 (bytes)
  Number of program headers:             9
  Size of section headers:               64 (bytes)
  Number of section headers:             32
  Section header string table index: 31
```

2.8.4 ファイルの属性: stat

`stat` コマンドを利用すると、ファイルの属性を表示することができます:

```
tux@mercury:~> stat /etc/profile
  File: '/etc/profile'
  Size: 9662      Blocks: 24      IO Block: 4096   regular file
Device: 802h/2050d Inode: 132349  Links: 1
Access: (0644/-rw-r--r--)  Uid: (  0/   root)   Gid: (  0/   root)
Access: 2009-03-20 07:51:17.000000000 +0100
```

Modify: 2009-01-08 19:21:14.000000000 +0100

Change: 2009-03-18 12:55:31.000000000 +0100

--filesystem オプションを指定すると、指定したファイルが 配置されているファイルシステムに関する属性情報を表示します:

```
tux@mercury:~> stat /etc/profile --file-system
File: "/etc/profile"
ID: d4fb76e70b4d1746 Namelen: 255      Type: ext2/ext3
Block size: 4096      Fundamental block size: 4096
Blocks: Total: 2581445   Free: 1717327   Available: 1586197
Inodes: Total: 655776    Free: 490312
```

2.9 ユーザ情報

2.9.1 ユーザのファイルアクセス監視: fuser

どのプロセスやユーザが特定のファイルにアクセスしているのかを確認 できれば便利です。たとえば /mnt にマウントされた ファイルシステムのマウントを解除したい場合、umount コマンドが "device is busy." としてマウントを解除できないことがあります。このような場合は、fuser コマンドを利用して、どのプロセスがデバイスを使用しているのかを確認することができます:

```
tux@mercury:~> fuser -v /mnt/*

          USER          PID ACCESS COMMAND
/mnt/notes.txt    tux      26597 f....  less
```

上記の例では他の端末から less プロセスを終了させると、ファイルシステムのマウント解除を行なうことができるようになります。なお -k オプションを指定すると、fuser は該当するファイルにアクセスしているプロセスに対し、プロセスの終了を行なうようになります。

2.9.2 ユーザの現状表示: w

w コマンドを利用すると、システムに誰がログインしているのかと、そのユーザが何をしているのかを表示することができます。たとえば下記ようになります:

```
tux@mercury:~> w
 14:58:43 up 1 day,  1:21,  2 users,  load average: 0.00, 0.00, 0.00
USER      TTY      LOGIN@  IDLE   JCPU   PCPU   WHAT
```

```
tux      :0      12:25  ?xdm?  1:23   0.12s /bin/sh /usr/bin/startkde
root    pts/4    14:13   0.00s  0.06s  0.00s w
```

他のシステムのユーザが遠隔 (リモート) からログインしている場合、`-f` オプションを指定すると、どのコンピュータから ログインしているのかが表示されるようになります。

2.10 日付と時刻

2.10.1 経過時間の計測: `time`

`time` ユーティリティを利用すると、指定したコマンドを実行する際に消費された時間を表示することができます。このユーティリティは シェルの内蔵機能で提供されているほか、プログラム (`/usr/bin/time`) としても提供されています。

```
tux@mercury:~> time find . > /dev/null
```

```
real    0m4.051s❶
user    0m0.042s❷
sys     0m0.205s❸
```

- ❶ コマンドを起動してから終了するまでにかかった実際の時間を示しています。
- ❷ `times` システムコールで報告された値で、ユーザ側の処理にかかった CPU 時間を示しています。
- ❸ `times` システムコールで報告された値で、システム側の処理にかかった CPU 時間を示しています。

2.11 データのグラフ化: `RRDtool`

この世界には計測可能なデータが数多く存在しています。たとえば温度の変化やお使いのコンピュータのネットワークインターフェイスが送受信したデータ量などがあります。`RRDtool` ではそのようなデータを保管し、カスタマイズ可能なグラフとして視覚化することができます。

`RRDtool` は多くの UNIX プラットフォームや Linux ディストリビューションに含まれています。もちろん `openSUSE®` でも同様です。`YaST` からインストールを行なうか、`root` になった状態で、下記のようなコマンドラインを入力してインストールしてください。


```
zypper install rrdtool
```

ヒント

それぞれ RRDtool には Perl, Python, Ruby, PHP の各バインディングが存在しています。これにより好きな言語で監視用のスクリプトを作成することができます。

2.11.1 RRDtool の動作

RRDtool は *Round Robin Database tool* (ラウンドロビン (循環) 型データベースツール) の省略形で、一定量のデータを取り扱うための仕組みです。データ行に始まりも終わりも存在しないような循環型のバッファを使用してデータを読み込みます。これをラウンドロビンデータベースと呼んでいます。

上述のとおり、RRDtool は時間が経過するごとに変化するようなデータを扱うよう設計されています。理想的には一定間隔でデータ (温度や速度など) を計測するような センサーを想定していて、このデータを一定の書式で出力する設計になっています。このようなデータであれば RRDtool が便利に利用でき、期待通りの出力を作成することができます。

場合によっては自動的かつ一定間隔で採取できないようなデータもあります。このような場合は RRDtool に与える前に事前の処理が必要で、条件によっては RRDtool を手動で動作させなければならない場合もあります。

下記では RRDtool の基本的な使い方を示しています。一般的な RRDtool の作業であるデータベースの *作成* と計測値による *更新*、および出力の *閲覧* をそれぞれ行なっています。

2.11.2 シンプルな実例

ここでは、Linux システムにおけるメモリ使用量について時間による変化を収集して 閲覧する仕組みを構築したい場合を想定します。より詳しく書くと、この例ではその時点での空きメモリ量を 4 秒おきに 40 秒間採取しています。この計測中、大量のシステムメモリを使用するプログラムが 3 つほど起動および終了しています。具体的には Firefox Web ブラウザと Evolution 電子メールクライアント、および Eclipse 開発フレームワークです。

2.11.2.1 データの収集

RRDtool はネットワークトラフィックの計測や可視化を行なう場合にもよく利用されています。このような場合は Simple Network Management Protocol (SNMP) を利用して行ないます。このプロトコルではそれぞれのネットワークデバイスに対して内部カウンタの値を問い合わせることができるため、RRDtool を利用してこれらの値を保存することができます。SNMP について詳しくは <http://www.net-snmp.org/> をお読みください。

今回の例は上記とは異なり、データを手動で採取する必要があります。下記にある `free_mem.sh` 支援スクリプトがメモリの空き容量について 現在の状態を読み取り、標準出力に書き込む作業を行ないます。

```
tux@mercury:~> cat free_mem.sh
INTERVAL=4
for steps in {1..10}
do
    DATE=`date +%s`
    FREEMEM=`free -b | grep "Mem" | awk '{ print $4 }'`
    sleep $INTERVAL
    echo "rrdtool update free_mem.rrd $DATE:$FREEMEM"
done
```

注目点

- 時間間隔が 4 秒に設定されていますが、これは `sleep` コマンドで実現しています。
- RRDtool は特殊な書式で時刻情報を受け入れます。具体的には *Unix 時刻* と呼ばれるもので、1970 年 1 月 1 日の深夜 00:00 からの経過秒数を表わしているものです。たとえば 1272907114 は 2010-05-03 17:18:34 を意味しています。
- メモリの空き容量情報は `free -b` を利用することでバイト単位で報告されます。ここではキロバイトのような おおまかな単位ではなく、バイト単位で正確な値を採取しています。
- `echo ...` のコマンドには、データベースファイル (`free_mem.rrd`) が指定されていて、ここから RRDtool のデータベースを更新するためのコマンドラインが書かれています。

`free_mem.sh` を実行すると、下記のような出力が現われます:

```
tux@mercury:~> sh free_mem.sh
rrdtool update free_mem.rrd 1272974835:1182994432
```

```
rrdtool update free_mem.rrd 1272974839:1162817536
rrdtool update free_mem.rrd 1272974843:1096269824
rrdtool update free_mem.rrd 1272974847:1034219520
rrdtool update free_mem.rrd 1272974851:909438976
rrdtool update free_mem.rrd 1272974855:832454656
rrdtool update free_mem.rrd 1272974859:829120512
rrdtool update free_mem.rrd 1272974863:1180377088
rrdtool update free_mem.rrd 1272974867:1179369472
rrdtool update free_mem.rrd 1272974871:1181806592
```

この出力はファイルに転送しておくのが便利です。下記のようにして行ないます。

```
sh free_mem.sh > free_mem_updates.log
```

2.11.2.2 データベースの作成

今回の設定で初期のラウンドロビンデータベースを作成するには、下記のコマンドを入力します:

```
rrdtool create free_mem.rrd --start 1272974834 --step=4 ¥
DS:memory:GAUGE:600:U:U RRA:AVERAGE:0.5:1:24
```

注目点

- このコマンドは `free_mem.rrd` という名前の ファイルを作成します。このファイルはラウンドロビン (循環) 型のデータベース で、計測値を保管するために使用するものです。
- `--start` オプションでは、データベースに追加されるデータの うち、最初の時間を Unix 時間形式で指定しています。この例では、`free_mem.sh` の最初の値 (1272974835) よりも小さい値を 設定しています。
- `--step` では、データベースに対してデータを提供する間隔 を秒単位で指定しています。
- `DS:memory:GAUGE:600:U:U` の部分はデータベースの データソースを指定しています。*memory* という名称で *gauge* というタイプのもので、2 つの更新の最長間隔は 600 秒に設定しています。計測値の *最小* と *最大* はそれぞれ未知の値 (U) に設定しています。
- `RRA:AVERAGE:0.5:1:24` はラウンドロビンアーカイブ (RRA) を作成するもので、*統合機能* (CF) を利用して データポイントの *平均値* を計算し、保管を行ないます。統合機能の 3 つのパラメータは、行の終わりに追加します。

何もエラーメッセージが表示されなければ、カレントディレクトリに `free_mem.rrd` データベースが作成されています:

```
tux@mercury:~> ls -l free_mem.rrd
-rw-r--r-- 1 tux users 776 May  5 12:50 free_mem.rrd
```

2.11.2.3 データベースの値の更新

データベースを作成したら、あとは計測データを書き込む作業です。2.11.2.1項「データの収集」(44 ページ) では既に `rrdtool update` コマンドを実行するためのファイル `free_mem_updates.log` を用意してあるので、これらのコマンドを利用してデータベースの値を更新することができます。

```
tux@mercury:~> sh free_mem_updates.log; ls -l free_mem.rrd
-rw-r--r-- 1 tux users 776 May  5 13:29 free_mem.rrd
```

上記のとおり、`free_mem.rrd` のサイズは更新前後で 変化することはありません。

2.11.2.4 計測値の閲覧

値を測定してデータベースを作成し、計測値を保管したら、そのデータベースを利用してデータを取り出したり、表示したりすることができますようになります。

データベースから全ての値を取り出すには、下記のコマンドを入力します:

```
tux@mercury:~> rrdtool fetch free_mem.rrd AVERAGE --start 1272974830 ¥
--end 1272974871
memory
1272974832: nan
1272974836: 1.1729059840e+09
1272974840: 1.1461806080e+09
1272974844: 1.0807572480e+09
1272974848: 1.0030243840e+09
1272974852: 8.9019289600e+08
1272974856: 8.3162112000e+08
1272974860: 9.1693465600e+08
1272974864: 1.1801251840e+09
1272974868: 1.1799787520e+09
1272974872: nan
```

注目点

- `AVERAGE` はデータベースから平均値を取り出すための キーワードです。これはデータソースを `AVERAGE 1` つしか定義していない (2.11.2.2項「データベースの作成」(45 ページ)) ため、その他の機能は利用できません。

- 出力の 1 行目にはデータソースの名前が出力されています。これは 2.11.2.2項「データベースの作成」(45 ページ) で設定したものです。
- 列の左側にはそれぞれ個別のポイントにおける時刻が表示されています。右側には科学表記での計測値の平均が表示されています。
- 最後の行にある nan は「not a number」(数値ではないもの) の意味です。データが存在しない場合などに出力されます。

あとはデータベース内にある値を利用してグラフを作成することができます:

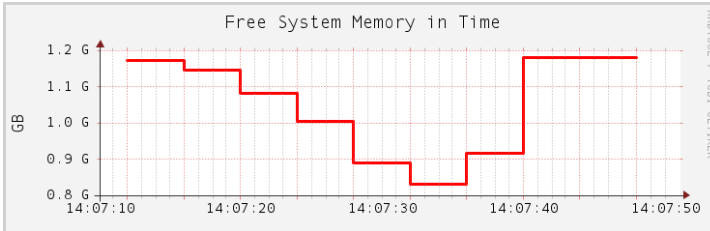
```
tux@mercury:~$ rrdtool graph free_mem.png ¥
--start 1272974830 ¥
--end 1272974871 ¥
--step=4 ¥
DEF:free_memory=free_mem.rrd:memory:AVERAGE ¥
LINE2:free_memory#FF0000 ¥
--vertical-label "GB" ¥
--title "Free System Memory in Time" ¥
--zoom 1.5 ¥
--x-grid SECOND:1:SECOND:4:SECOND:10:0:%X
```

注目点

- free_mem.png は作成するグラフのファイル名 を指定しています。
- --start と --end はグラフ内に 描画すべき時間範囲を指定しています。
- --step はグラフの時間間隔を秒単位で指定しています。
- DEF:... の部分は *free_memory* のデータ定義です。データは free_mem.rrd ファイル から読み込みを行ない、*memory* という名前を設定 します。また、*average* (平均値) を計算して出力します。これは、2.11.2.2項「データベースの作成」(45 ページ) で設定した とおり、それ以外の値を定義していないためです。
- LINE... の部分はグラフ内に描画する線の属性を設定して います。2 ピクセル幅で *free_memory* で定義した データを利用し、赤色で描画します。
- --vertical-label では y 軸に 表示するラベルを指定しています。また、--title では グラフ全体に対するラベル (つまりタイトル) を指定しています。
- --zoom はグラフの拡大率を指定しています。この値は ゼロより大きい値でなければなりません。

- `--x-grid` ではグラフ内におけるグリッド線とラベルの描画方法を指定しています。この例では 4 秒ごとにグリッド線を出力しています。ラベルは各グリッド線に対して 10 秒ごとに配置しています。

図 2.2 RRDtool で作成したグラフ例



2.11.3 さらなる情報

RRDtool には数多くのサブコマンドとコマンドラインオプションが存在し、非常に複雑なツールになっています。これらのうちのいくつかは理解しやすいものですが、要件に応じた出力や好みに沿った指定を行なうには、*学習* をしなければ習得できません。

基本的な情報だけを提供している RRDtool のマニュアルページ (`man 1 rrdtool`) 以外に、RRDtool homepage [<http://oss.oetiker.ch/rrdtool/>] (英語) についてもお読みになることをお勧めします。ここには `rrdtool` の詳細な文書 [<http://oss.oetiker.ch/rrdtool/doc/index.en.html>] が置かれていて、コマンドとサブコマンドの全てが収録されています。また、一般的な RRDtool の作業手順を示しているチュートリアル [<http://oss.oetiker.ch/rrdtool/tut/index.en.html>] もお勧めです。

また、ネットワークトラフィックの監視を行ないたい場合は、MRTG [<http://oss.oetiker.ch/mrtg/>] (英語) もお読みください。このツールは Multi Router Traffic Grapher の意味で、ネットワークデバイスに対する全ての動作状況をグラフ化することができる RRDtool よりも使いやすいツールです。

Nagios を利用した監視

Nagios は安定していてスケーラビリティに富んだ、拡張性の高いエンタープライズ用途のネットワーク／システム監視ツールです。管理者に対してネットワークとシステムの監視機能を提供し、HTTP, SMTP, POP3 のほか、ディスク使用率やプロセスの負荷などを監視することができます。もともと Nagios は Linux 上で動作するように設計されてきましたが、一般的な UNIX オペレーティングシステムでも動作します。この章では、Nagios のインストールと設定方法について説明しています。Nagios のプロジェクトについて、詳しくは <http://www.nagios.org/> をお読みください。

3.1 Nagios の機能

Nagios の主要な機能は下記のとおりです:

- ネットワークサービスの監視 (SMTP, POP3, SMTP, NNTP など)
- ホスト資源の監視 (プロセスの負荷, ディスク使用率など)
- シンプルなプラグイン機能による、さらなるサービスチェック機能の追加
- 冗長構成の Nagios サーバへの対応

3.2 Nagios のインストール

Nagios は zypper コマンドまたは YaST でインストールします。

パッケージのインストール方法は下記のいずれかをお読みください:

- 項「zypper の使用」(第9章 コマンドラインツールを利用したソフトウェア管理, ↑ スタートアップ)
- 項「パッケージやパターンのインストールと削除」(第5章 ソフトウェアのインストールと削除, ↑ スタートアップ)

いずれの手順でも、nagios と nagios-www のパッケージをそれぞれ インストールしてください。後者の RPM パッケージには Nagios に対する Web インターフェイスが含まれていて、たとえばサービスの状態や問題の履歴などを Web 経由で閲覧することができます。なお、後者のパッケージは必須というわけでは ありません。

Nagios はモジュールを利用する設計になっているため、チェックプラグインを 追加することで、特定のサービスが利用可能かどうかを調べることができます。nagios-plugin パッケージでは、そのような出来合いの プラグインが収録されていますので、インストールしておくことをお勧めします。もちろん独自のチェックプラグインを作成することもできます。

3.3 Nagios の設定ファイル

Nagios には、下記のような設定ファイルがあります:

/etc/nagios/nagios.cfg

Nagios の根幹部分の設定ファイルで、Nagios の動作を決定する各種の ディレクティブが書かれています。詳しい文書については、http://nagios.sourceforge.net/docs/3_0/configmain.html をお読みください。

/etc/nagios/resource.cfg

すべての Nagios プラグインに対するパスを記述します (既定では /usr/lib/nagios/plugins が 書かれています)。

/etc/nagios/command.cfg

サービスが利用可能かどうかを判断するためのプログラム定義や、電子メールによる通知を送信するためのコマンド定義を記述します。

/etc/nagios/cgi.cfg

Nagios Web インターフェイス関連のオプションを記述します。

/etc/nagios/objects/

オブジェクト定義ファイルを含むディレクトリです。詳しくは 3.3.1 項「オブジェクト定義ファイル」(51 ページ) をお読みください。

3.3.1 オブジェクト定義ファイル

上述のような設定ファイルに加え、Nagios では非常に柔軟で高度なカスタマイズのできる、**オブジェクト定義ファイル** と呼ばれる設定ファイルが 存在します。これらの設定ファイルでは下記のようなオブジェクトを定義することができるため、非常に重要な存在となっています：

- ホスト
- サービス
- 連絡先

オブジェクトは容易に拡張できるようになっていることから、より柔軟に 対応できるようになっています。たとえば 1 つのサービスだけが動作している ホストを管理している場合を想定してみてください。同じホストに対して別の サービスをインストールし、そのサービスについても監視を行ないたい場合、新しいほうのサービスをオブジェクトとして追加し、それをホストオブジェクトに 割り当てれば、面倒な手間をかけることなく新しいサービスを監視できるようになります。

インストールが終わっていれば、Nagios ではオブジェクト定義の設定ファイルについて、既定のテンプレートが提供されます。テンプレートは /etc/nagios/objects 内に存在していますので、下記のようにホストやサービス、連絡先を追加することができます：

例 3.1 ホストオブジェクトの定義

```
define host {
    name                SRV1
    host_name            SRV1
    address              192.168.0.1
    use                  generic-host
    check_period         24x7
    check_interval       5
    retry_interval       1
    max_check_attempts   10
    notification_period   workhours
    notification_interval 120
    notification_options d,u,r
}
```

host_name オプションでは、ホストに対する監視時の 表示名を指定しています。また address には、この ホストの IP アドレスを指定しています。use ステートメントでは、generic-host という名称の他の設定値を継承する指定が 書かれています。さらに check_period では、24x7 (24 時間 7 日、つまり年中無休) で監視することを指定しています。また、check_interval では 5 分間隔で監視する 設定が書かれているほか、retry_interval では 再試行の間隔が 1 分間隔であることを指定しています。また、Nagios では 正当性が確認できなかったチェック に対して、複数回の再試行を行ないます。再試行の回数を指定する項目として、max_check_attempts ディレクティブが用意されています。また、notification で 始まるすべての設定フラグは、Nagios が監視対象のサービスに異常を検出した 場合に、どのような処理を行なうべきかを指定します。上記のホスト定義では、Nagios は日中の時間帯に管理者に対してのみ通知を行ないます。時間帯の 設定は notification_period で行ないます。また notification_interval によると、通知は 2 時間ごとに送信されます。さらに notification_options では、4 種類 のフラグ d, u, r, n を設定することができます。ここでは管理者に対してどのような通知を行なうのかを指定します。d はサービスが動作していない (down) 状態を、u はネットワーク経由で到達不能な状態 (unreachable) 、そして r は それらの状態からの復旧 (recoveries) について、それぞれ 通知を行なうように指定します。なお、n は 通知を全く行なわないようにするフラグです。

例 3.2 サービスオブジェクトの定義

```
define service {
    use                generic-service
    host_name          SRV1
    service_description PING
    contact_groups      router-admins
    check_command       check_ping!100.0,20%!500.0,60%
}
```

最初の設定ディレクティブである use では、Nagios に対して generic-service のテンプレートから 設定を引き継ぐように指定しています。次の host_name はサービスを割り当てる先のホストオブジェクトを指定しています。ホスト 自体の設定は、ホストオブジェクトとして設定してください。また、service_description ではサービスに対する説明文を 設定することができます。上記の例では単純に PING とだけ 書かれています。contact_groups オプションでは、サービスが動作していない場合の連絡先となるグループを指定します。グループと その中のメンバーは、後述する連絡先グループオブジェクトで設定します。最後に check_command では、そのサービスが利用 可能かどうかを調べるプログラムを設定します。

例 3.3 連絡先と連絡先グループの定義

```
define contact {
    contact_name      admins
```

```

use                generic-contact
alias              Nagios Admin
email              nagios@localhost
}

define contactgroup {
    contactgroup_name router-admins
    alias              Administrators
    members            admins
}

```

上記の例では、contact を利用した直接的な連絡先 設定と、それを包含する contactgroup による 連絡先グループを示しています。contact では、サービスが利用できない場合に利用する電子メールアドレスと、そのユーザの 名前を設定します。通常、ここは責任を持つ管理者を設定します。use では、generic-contact という名称で設定された 既定値を引き継ぐことを指定しています。

Nagios のすべてのオブジェクトについて、詳しくは下記をお読みください: http://nagios.sourceforge.net/docs/3_0/objectdefinitions.html.

3.4 Nagios の設定

リモートのサービスやリモートホストの資源など、様々な項目を監視するにあたっての Nagios の設定方法を説明します。

3.4.1 Nagios を利用したリモートサービスの監視

この章では、Nagios を利用してリモートのサービスを監視するための 方法を示しています。具体的には下記のようにして実施してください:

手順 3.1 Nagios を利用したリモートの HTTP サービスの監視

- 1 mkdir コマンドを利用し、/etc/nagios/objects 内にディレクトリを作成します。作成するディレクトリ名は任意の名前でかまいません。
- 2 /etc/nagios/nagios.conf ファイルを開き、最初の段階で作成しておいたディレクトリを cfg_dir (設定ディレクトリ) 内に設定します。
- 3 最初の段階で作成した設定ディレクトリに移動し、それぞれ新しい ファイルを作成します: hosts.cfg, services.cfg, contacts.cfg

4 hosts.cfg ファイル内にホストオブジェクトを記述します:

```
define host {
    name                host.name.com
    host_name           host.name.com
    address             192.168.0.1
    use                 generic-host
    check_period        24x7
    check_interval      5
    retry_interval      1
    max_check_attempts  10
    contact_groups      admins
    notification_interval 60
    notification_options d,u,r
}
```

5 services.cfg 内にサービスオブジェクトを記述します:

```
define service {
    use                 generic-service
    host_name           host.name.com
    service_description HTTP
    contact_groups      router-admins
    check_command       check_http
}
```

6 contacts.cfg 内に連絡先および連絡先 グループオブジェクトを記述します:

```
define contact {
    contact_name        max-mustermann
    use                 generic-contact
    alias               Webserver Administrator
    email               mmustermann@localhost
}

define contactgroup {
    contactgroup_name   admins
    alias               Administrators
    members             max-mustermann
}
```

7 rcnagios restart を実行し、Nagios を再起動します。

8 cat /var/log/nagios/nagios.log を実行し、下記の ような出力があることを確認します:

```
[1242115343] Nagios 3.0.6 starting... (PID=10915)
[1242115343] Local time is Tue May 12 10:02:23 CEST 2009
[1242115343] LOG VERSION: 2.0
```

[1242115343] Finished daemonizing... (New PID=10916)

異なるリモートサービスを監視したい場合は、ステップ 5 (54 ページ) のステップで `check_command` の項目を調整します。利用可能なすべてのチェックプログラムについて、一覧を表示するには、`ls /usr/lib/nagios/plugins/check_*` を実行してください。

何らかのエラーが発生した場合は、3.5項「トラブルシューティング」(57 ページ) をお読みください。

3.4.2 Nagios を利用したリモートホストの資源監視

この章では、Nagios を利用してリモートホストの資源を監視するための手順を説明しています。

Nagios サーバ上で下記の手順を実施します:

手順 3.2 Nagios を利用したリモートホストの資源監視 (サーバ)

1 `nagios-nscd` パッケージを インストールします (たとえば `zypper in nagios-nscd` など)。

2 `/etc/nagios/nagios.cfg` ファイル内で、下記の設定を 行います:

```
check_external_commands=1
accept_passive_service_checks=1
accept_passive_host_checks=1
command_file=/var/spool/nagios/nagios.cmd
```

3 `/etc/nagios/nscd.conf` ファイル内の `command_file` オプションについて、`/etc/nagios/nagios.conf` と同じファイルを指定します。

4 他のホストやサービスのオブジェクトを追加します:

```
define host {
    name                foobar
    host_name            foobar
    address              10.10.4.234
    use                  generic-host
    check_period         24x7
    check_interval       0
    retry_interval       1
    max_check_attempts   1
    active_checks_enabled 0
```

```

passive_checks_enabled      1
contact_groups              router-admins
notification_interval       60
notification_options        d,u,r
}

define service {
    use                      generic-service
    host_name                foobar
    service_description      diskcheck
    active_checks_enabled    0
    passive_checks_enabled   1
    contact_groups           router-admins
    check_command             check_ping
}

```

5 rcnagios restart と rcnsca restart コマンドをそれぞれ実行します。

監視対象のクライアント側では、下記の手順を実施します:

手順 3.3 Nagios を利用したリモートホストの資源監視 (クライアント)

- 1 監視対象 (監視を受ける) のホストに対して、nagios-nsca-client パッケージをインストールします。
- 2 下記のようにして、テスト用のスクリプトを作成します (たとえばディスクの 使用率をチェックするスクリプトなど):

```

#!/bin/bash
NAGIOS_SERVER=10.10.4.166
THIS_HOST=foobar

#
# ここに独自のテストアルゴリズムを記述します
#

# 成功時に実行するもの:
echo "$THIS_HOST;diskcheck;0;OK: test ok" ¥
    | send_nsca -H $NAGIOS_SERVER -p 5667 -c /etc/nagios/send_nsca.cfg -d ";"

# 警告時に実行するもの:
echo "$THIS_HOST;diskcheck;1;Warning: test warning" ¥
    | send_nsca -H $NAGIOS_SERVER -p 5667 -c /etc/nagios/send_nsca.cfg -d ";"

# 失敗時に実行するもの:
echo "$THIS_HOST;diskcheck;2;CRITICAL: test critical" ¥
    | send_nsca -H $NAGIOS_SERVER -p 5667 -c /etc/nagios/send_nsca.cfg -d ";"

```

- 3 あとは crontab -e コマンドを利用して、新しい cron 項目を作成します。たとえば下記のようになります:

`*/5 * * * * /directory/to/check/program/check_diskusage`

3.5 トラブルシューティング

Error: ABC 'XYZ' specified in ... '...' is not defined anywhere!

すべてのオブジェクトが正しく定義されていることを確認してください。また、スペルミスには注意してください。

(Return code of 127 is out of bounds - plugin may be missing)

nagios-plugins パッケージが インストールされているかどうか確認してください。

電子メールによる通知が動作しない

postfix や exim などの メールサーバがインストールされていて、正しく設定されているかどうかを確認してください。メールサーバが正しく動作しているかどうかを確認するには、`echo "Mail Server Test!" | mail foo@bar.com` のように 入力します。このコマンドは `foo@bar.com` 宛にメールを送信します。このメールが 正しく配信されれば、お使いのメールサーバは問題なく動作していることになります。配信されない場合は、メールサーバのログファイルを確認してください。

3.6 さらなる情報

Nagios の完全なドキュメンテーション

http://nagios.sourceforge.net/docs/3_0/toc.html

オブジェクト設定の概要

http://nagios.sourceforge.net/docs/3_0/configobject.html

オブジェクトの定義について

http://nagios.sourceforge.net/docs/3_0/objectdefinitions.html

Nagios のプラグイン

http://nagios.sourceforge.net/docs/3_0/plugins.html

システムログファイルの分析と管理

システムログファイルの分析作業は、システムを分析するにあたって最も重要な作業のうちの 1 つです。実際、システムログファイルの分析作業は、システムのメンテナンスやトラブル解析を行なう際、最初にやっておかなければならない作業です。openSUSE では、システムで発生した各種の事象について、ほぼすべての詳細を自動的に記録するようになっています。通常、システムログファイルは単なるテキスト形式のファイルに記録されるため、エディタやページャなどを利用して簡単に閲覧することができます。これらはスクリプトなどからも処理できる仕組みになっていて、内容文でフィルタする作業も簡単に行なうことができます。

4.1 /var/log/ 内にあるシステムログファイル

システムログファイルは、必ず /var/log ディレクトリ以下に配置されます。下記の表では、既定の手順でインストールを行なった場合に openSUSE から提供される、すべてのシステムログファイルに関する概要を示しています。お使いのマシンのインストール形態によっては、/var/log ディレクトリ内に、ここでは説明していない他のサービスやアプリケーションからのログが存在する場合があります。また、下記で説明しているファイルやディレクトリは、関連するアプリケーションがインストールされている場合にのみ使用される「プレースホルダ」を示している場合があります。また、多くのログファイルは root でないと閲覧できません。

acpid

ACPI (Advanced Configuration and Power Interface) のイベントをユーザースペース側のプログラムに通知するデーモン、acpid (acpid) が出力するログ

ファイルです。acpid では、STDOUT や STDERR と同様に、すべての動作をログファイルに記録します。

apparmor

AppArmor のログファイルです。AppArmor について、詳しくは パート「AppArmor を利用した権利制限」(↑セキュリティガイド)をお読みください。

audit

監査フレームワークが提供するログファイルです。

boot.msg

システムの起動処理時のログファイルです。このファイルには、カーネルが出力した起動時のメッセージのほか、起動処理時に実行された起動スクリプトや開始されたサービスからのメッセージがすべて記録されています。

お使いのハードウェアが正しく初期化されているかどうかや、すべてのサービスが正しく起動できているかどうかを確認するには、このファイルをお読みください。

boot.omsg

システムのシャットダウン処理時のメッセージファイルです。このファイルには、直近のシャットダウンや再起動で出力された、すべてのメッセージが記録されています。

ConsoleKit/*

ConsoleKit デーモン (ユーザのログインのほか、コンピュータをどのように利用したのかを追跡する デーモン) のログファイルです。

cups/

cups (common UNIX printing system) (cups) が記録するアクセスログとエラーログが保存されます。

faillog

すべてのログイン失敗事象を記録するデータベースファイルです。このファイルを閲覧するには、faillog コマンドをお使いください。詳しくは man 8 faillogをお読みください。

firewall

ファイアウォールのログファイルです。

gdm/*

GNOME ディスプレイマネージャのログファイルです。

krb5

Kerberos ネットワーク認証システムのログファイルです。

lastlog

lastlog ファイルは、各ユーザの最終ログインを記録するデータベースです。このファイルを閲覧するには、lastlog コマンドをお使いください。詳しくは man 8 lastlog をお読みください。

localmessages

DHCP クライアントのログなど、起動スクリプトからのメッセージが記録されるログファイルです。

mail*

メールサーバ (postfix, sendmail) のログファイルです。

messages

すべてのカーネルメッセージやシステムメッセージが記録される、既定のログファイルです。何らかの問題が発生した場合、/var/log/warn とともに最初に確認しておくべきログファイルです。

NetworkManager

NetworkManager のログファイルです。

news/*

news サーバのログファイルです。

ntp

NTP (Network Time Protocol) デーモン (ntpd) のログファイルです。

pk_backend_zypp

PackageKit (と libzypp バックエンド) のログファイルです。

puppet/*

データセンター自動化ツールである puppet のログファイルです。

samba/*

Windows SMB/CIFS ファイルサーバである samba のログファイルです。

SaX.log

SaX2 (SUSE advanced X11 configuration tool) のログファイルです。

scpm

SCPM (system configuration profile management) (scpm) のログファイルです。

warn

すべてのシステム警告とエラーが記録されるログファイルです。何らかの問題が発生した場合、`/var/log/messages` とともに最初に確認しておくべきログファイルです。

wtmp

すべてのログイン／ログアウトやランレベルの変更、およびリモート接続の動作を記録するデータベースファイルです。このファイルを閲覧するには、`last` コマンドをお使いください。詳しくは `man 1 last` をお読みください。

xinetd.log

xinetd (extended Internet services daemon) (xinetd) のログファイルです。

Xorg.0.log

X の起動時のログファイルです。X の起動に失敗するような場合、このファイルをお読みください。それぞれ古いログファイルは `Xorg.?.log` として保存されます。

YaST2/*

すべての YaST ログファイルが保管されるディレクトリです。

zypp/*

libzypp のログファイルが保管されるディレクトリです。これらのファイルを利用することで、パッケージのインストール履歴を確認することができます。

zypper.log

コマンドラインインストーラである zypper のログファイルです。

4.2 ログファイルの閲覧と分析

ログファイルは、お使いのテキストエディタで開くだけで閲覧することができます。それ以外にも、YaST コントロールセンターから *その他 > システムログ* を選択すると、`/var/log/messages` を閲覧することができます。

テキストコンソール内でログファイルを閲覧する場合は、`less` や `more` を利用するのが便利です。これ以外にも、`head` や `tail` を利用することで、ログファイルの冒頭や末尾を表示することができます。ログファイルに追記される内容をリアルタイムに閲覧するには、`tail -f` をお使いください。これらのツールについての詳細は、各マニュアルページをお読みください。

また、ログファイル内で文字列や正規表現による検索を行ないたい場合は `grep` を利用します。ログファイルを自動的に処理したり、書き換えたりしたい場合は、`awk` が便利でしょう。

4.3 logrotate を利用したログファイルの管理

`/var/log` ディレクトリ内に記録されるログファイルは 日々追記されるため、知らないうちに肥大化してしまうようなことがよくあります。logrotate コマンドはそのような巨大なログファイルを 処理するツールで、これらのファイルの管理を容易にし、肥大化を制御するための 仕組みが備わっています。具体的にはログファイルの自動的なローテーション (切り替え) や削除、圧縮やメール送付などに対応しています。また、ログファイルは 定期的 (日次、週次、月次) に処理させることができるほか、指定したサイズを 超えた場合などを指定することができます。

logrotate は通常、日次の cron ジョブとして実行します。また、サイズ超過によるものや logrotate を 1 日に複数回 実施したり、`--force` オプションを指定したりしない限り、1 日 1 回以上のログファイル処理は行なわれません。

logrotate の中枢となる設定ファイルは `/etc/logrotate.conf` です。ログファイルを生成するような プログラム (たとえば `apache2` など) の場合、そのシステムパッケージには `/etc/logrotate.d/` 内に配置される独自の設定ファイルが含まれています。`/etc/logrotate.d/` ディレクトリは、`/etc/logrotate.conf` から読み込むように指定しています。

例 4.1 `/etc/logrotate.conf` の設定例

```
# 詳細は "man logrotate" をお読みください。
# ログファイルを毎週ローテーション (切り替え) します
weekly

# 4 週分までの過去ログを保存します
rotate 4

# 古いログファイルをローテーションした場合、新しい (何も無い) ファイルを作成します
create

# ローテーションしたファイルには、接尾辞として日付を付与します
dateext

# ログファイルを圧縮したい場合、下記のコメントを外してください
#compress
```

```
# gzip やその他の圧縮方法を使用したい場合、下記の行をコメントアウトしてください
scompresscmd /usr/bin/bzip2
uncompresscmd /usr/bin/bunzip2

# 各種の RPM パッケージでは、下記のディレクトリ内にローテーション情報を
# 配置します
include /etc/logrotate.d
```

重要

create オプションを利用する場合は、/etc/permissions* ファイル内で指定されるファイルのパーミッション (アクセス権) や所有者情報にご注意ください。これらの設定を修正する場合は、矛盾がないことをよくご確認ください。

logrotate は cron の設定ファイル /etc/cron.daily/logrotate 内で設定され呼び出されます。直近でどのファイルがローテーションされたのかを知るには、/var/lib/logrotate.status をお読みください。

4.4 logwatch を利用したログファイルの監視

logwatch はカスタマイズの可能なプラグイン型ログ監視 スクリプトです。システムログを処理して最も重要な情報だけを取り出し、人間にとって読みやすい形式で表示します。logwatch コマンドを利用するには、logwatch パッケージをインストールしてください。

logwatch はコマンドラインから実行することで、その時点の レポートを生成することができますほか、cron を介して実行することで、カスタムな レポートを生成することができます。レポートは画面上に表示させることができるほか、ファイルへの保存や特定アドレスへのメール送信を行なうことができます。後者は特に cron での自動生成時に有用な機能です。

コマンドラインの文法は簡単な仕組みです。基本的には logwatch に続いて生成するレポートに記載するサービスの種類や時間範囲、詳細レベルなどを指定します:

```
# 昨日以降のすべてのカーネルメッセージについて詳細なレポートを生成する
logwatch --service kernel --detail High --range Yesterday --print

# すべての sshd 関連の記録イベントについて、詳細レベルを低くしてレポートを生成する (アーカイブされたログを含む)
Low detail report on all sshd events recorded (incl. archived logs)
logwatch --service sshd --detail Low --range All --archives --print
```

```
# 5 月 5 日から 5 月 7 日までに生成されたすべての smartd メッセージについて、 root@localhost 宛にメール  
でレポートを送信する  
logwatch --service smartd --range 'between 5/5/2005 and 5/7/2005' ¥  
--mailto root@localhost --print
```

--range オプションでは複雑な書き方をすることもできます。詳しくは `logwatch --range help` を実行してください。また、問い合わせ可能なすべてのサービスについて一覧を表示するには、下記の コマンドを実行します:

```
ls /usr/share/logwatch/default.conf/services/ | sed 's/¥.conf//g'
```

logwatch は、かなり細かい範囲にまで設定を行なうことができます。ですが、ほとんどの場合は既定の設定のみで十分です。既定の設定ファイルは `/usr/share/logwatch/default.conf/` にあります。この ファイルは直接編集したりしないでください。これはパッケージに含まれるファイルであり、今後のソフトウェア更新で上書きされてしまう可能性があるためです。その 代わり、独自の設定ファイルを `/etc/logwatch/conf/` 内に 作成してお使いください (上述のファイルを雛形として使用してもかまいません)。logwatch のカスタマイズ方法について、手順などの HOWTO が `/usr/share/doc/packages/logwatch/HOWTO-Customize-LogWatch` 内にあります。また、下記のような設定ファイルもあります:

`logwatch.conf`

メインの設定ファイルです。既定のバージョンは細かくコメントが書かれています。それぞれの設定オプションは、コマンドラインから上書きすることができます。

`ignore.conf`

logwatch で無視されるべきすべての行を記述します。

`services/*.conf`

このサービスディレクトリには、レポートを生成するための各サービスに関する設定ファイルが配置されます。

`logfiles/*.conf`

各サービスについて処理されるべきログファイルの仕様が書かれています。

4.5 システムログへの記録コマンド logger

logger はシステムログに対して、記録を行なうための ツールです。これはシェルのコマンドインターフェイスから、syslog(3) システムログモジュールへのインターフェ

イスを提供するものです。たとえば 下記のように実行すると、`/var/log/messages` 内に それが記録されます:

```
logger -t Test "This messages comes from $USER"
```

現在のユーザ名とホスト名に依存しますが、`/var/log/messages` ファイルには下記のような行が 記録されます:

```
Sep 28 13:09:31 venus Test: This messages comes from tux
```


パート III. カーネル監視

SystemTap—システムデータのフィルタと分析

SystemTap はコマンドラインインターフェイスとスクリプト言語を提供し、稼働中の Linux システム、特にカーネルについて、詳しい動作を調査することができる ソフトウェアです。SystemTap のスクリプトは SystemTap スクリプト言語で記述する仕組みになっていて、それらは C 言語のコードとしてコンパイルされ、カーネル モジュールの形に構築されて組み込まれます。スクリプトはデータを取り出して フィルタを設定し、収集するようになっているため、複雑な性能問題や機能問題を 診断することができます。SystemTap は netstat, ps, top, iostat などのツールが出力する のに似た形式で情報を提供しますが、収集された情報に対してフィルタや分析を 追加することができるという点が異なります。

5.1 考え方の概要

SystemTap スクリプトを起動すると、SystemTap セッションが開始されます。また、スクリプトの実行が許可されるまでの間には、スクリプトをカーネルモジュールとしてコンパイルし、読み込まれる手順が行なわれます。以前に対象のスクリプトを 実行済みで、構成に変更がない場合 (構成にはたとえば、コンパイラのバージョンや カーネルのバージョン、ライブラリパスやスクリプトの内容が含まれます) は、SystemTap はコンパイルし直したりすることはせず、SystemTap のキャッシュ (~/.systemtap) 内に含まれる *.c と *.ko を利用して実行します。読み込まれたモジュールは、tap の実行が終わると読み込み解除されます。動作例として、5.2項「インストールと設定」(72 ページ) に書かれている 動作テストをお読みください。

5.1.1 SystemTap のスクリプト

SystemTap の使用は、主に SystemTap のスクリプト (*.stp) を使用することで行ないます。スクリプトでは SystemTap がどのような種類の情報を収集し、その収集した情報に対して何かを行なうのかを指し示します。スクリプトは、AWK や C 言語に似た SystemTap スクリプト言語で記述します。言語の定義について、詳しくは <http://sourceware.org/systemtap/langref/> をお読みください。

SystemTap スクリプトの主な考え方は、イベントに対して命名を行ない、それらをハンドラに渡す、という流れを構築するということです。SystemTap がスクリプトを実行すると、特定のイベントに対して監視を行ないます。イベントが発生すると、Linux カーネルはサブルーチンとしてハンドラを動かし、元の実行に戻ります。そのため、イベントはハンドラを動作させるためのトリガー (きっかけ) という位置づけになります。ハンドラでは、指定したデータを指定した形式で記録または表示させることができます。

なお、SystemTap 言語ではごく少数のデータ型 (整数、文字列、およびそれらの連想配列) を使用し、完全な構造制御 (ブロック、条件分岐、ループ、関数) を扱うことができます。また、簡易な文章構造 (セミコロンによる区切りは任意) になっているほか、詳細な宣言を行なう必要はありません (データ型は自動的に推測され、チェックされます)。

SystemTap のスクリプトやそれらの文法について、詳しくは 5.3 項「スクリプトの文法」(74 ページ) のほか、stapprobes や stapfuncs のマニュアルページをお読みください。マニュアルページは、systemtap-docs パッケージに含まれています。

5.1.2 tap セット

tap セットとは事前に作成された探査用のライブラリで、SystemTap スクリプト内から使用することができる関数群を含んでいます。ユーザが SystemTap スクリプトを実行すると、SystemTap はスクリプトの探査イベントやハンドラが、tap セットライブラリ内に存在するかどうかを確認します。その後 SystemTap はスクリプトが C 言語のコードに変換される前に、それら必要な探査や関数を読み込みます。SystemTap スクリプト自身と同様に、tap セットのファイル名には *.stp という拡張子が付けられています。

ただし SystemTap スクリプトとは異なり、tap セットは直接実行するためのものではありません。他のスクリプトから定義を引き出すための書庫として存在しています。そのため tap セットライブラリは、イベントや関数を簡単に作成できるようにする

ための抽象層と言えます。tap セットはイベントを 指定したいユーザに対して、各種の便利な関数の別名を規定しています。たとえばバージョンごとに異なるような特定のカーネル関数に対して、わかりやすい別名が用意されていたりします。

5.1.3 コマンドと権限

SystemTap でもっとも使用されるコマンドは、stap と staprun の 2 つです。これらは root から実行するか、もしくは stapdev または stapusr のメンバーのユーザから 実行する必要があります。

stap

SystemTap のフロントエンドです。SystemTap のスクリプトを、ファイルや標準入力 から受け付けて実行することができます。スクリプトは C 言語のソースコード に変換されてコンパイルされ、実行中の Linux カーネル内にモジュールとして 読み込まれます。その後、必要なシステムトレースや探査機能が実施されます。

staprun

SystemTap のバックエンドです。SystemTap のフロントエンドが提供したカーネル モジュールを、読み込んだり解放したりします。

それぞれのコマンドについてオプションの一覧を表示するには、コマンドに --help オプションを付けて実行してください。また、詳しい説明 については、それぞれ stap と staprun のマニュアルページをお読みください。

また、SystemTap を実行するだけのユーザに対して、root の権限を付与せずに済ませる方法として、それぞれ下記に示す SystemTap 用のグループに所属させる方法があります。これらはいずれも openSUSE の既定の設定では用意されていませんが、下記の グループを作成して適切な権限を与えることで、上記を実現することができます。

stapdev

このグループのメンバーの場合、stap コマンドを利用して SystemTap のスクリプトを実行することができるほか、staprun コマンドで SystemTap の仲介モジュールを実行することができます。stap の実行には、カーネルモジュールのコンパイルや カーネルへの読み込みも含まれることになりますので、実質的には root の権限を持つことにもなります。

stapusr

このグループのメンバーの場合、staprun を利用して SystemTap の仲介モジュールを実行することだけが許可されます。これに加えて、/lib/

modules/カーネルバージョン/systemtap/ ディレクトリ以下にあるモジュールを実行することもできます。このディレクトリは root が所有者に設定されていない必要のないもので、root だけが書き込むことができます。

5.1.4 重要なファイルとディレクトリ

下記の一覧では、SystemTap における主なファイルとディレクトリについて、概要を説明しています。

/lib/modules/カーネルバージョン/systemtap/
SystemTap の仲介モジュールを保持しています。

/usr/share/systemtap/tapset/
tap セットの標準ライブラリを保持しています。

/usr/share/doc/packages/systemtap/examples
様々な目的で使用する SystemTap スクリプト例が多数収録されています。このディレクトリは、systemtap-docs パッケージをインストールした場合のみ利用できます。

~/.systemtap/cache
SystemTap ファイルのキャッシュを保持するデータディレクトリです。

/tmp/stap*
SystemTap ファイルの一時ディレクトリです。C 言語へ変換された後のファイルや、カーネルオブジェクトなどが配置されます。

5.2 インストールと設定

SystemTap はカーネルからの情報を 必要とするため、SystemTap パッケージに加えていくつかのカーネル関連パッケージを インストールしなければなりません。カーネル関連パッケージは SystemTap の探索対象の カーネルバージョンや種類によって異なるものであるため、それらが正しく一致しているパッケージをインストールしてください (下記の表では * としてバージョンや種類を示しています)。

重要: デバッグ情報付きパッケージのリポジトリ

お使いのシステムでオンライン更新を利用している場合は、openSUSE 12.2 用のオンラインインストールリポジトリ *-Debuginfo-Updates 内に「デバッグ情報」

(debuginfo) パッケージが存在しています。YaST を利用し、このリポジトリを有効に設定してください。

従来型の利用形態の場合は、下記のパッケージをインストールしてください (YaST または zypper でインストールを行なうことができます)。

- systemtap
- systemtap-server
- systemtap-docs (任意)
- kernel-*-base
- kernel-*-debuginfo
- kernel-*-devel
- kernel-source-*
- gcc

各種の目的でマニュアルページを読んだりサンプルの SystemTap スクリプト集を参照したりしたい場合は、systemtap-docs パッケージをインストールしてください。

お使いのマシンに必要な全てのパッケージがインストールされていて、SystemTap を利用できる状態にあるかどうかを確認するには、root で下記のコマンドを実行します。

```
stap -v -e 'probe vfs.read {printf("read performed¥n"); exit();}'
```

上記のコマンドを実行するとスクリプトが起動し、現在使用しているカーネルについてチェックを行ないます。下記のように出力されれば、SystemTap を利用する準備ができて いると判断できます:

```
Pass ❶: parsed user script and 59 library script(s) in 80usr/0sys/214real ms.
Pass ❷: analyzed script: 1 probe(s), 11 function(s), 2 embed(s), 1 global(s) in
140usr/20sys/412real ms.
Pass ❸: translated to C into
"/tmp/stapDwEk76/stap_1856e21ea1c246da85ad8c66b4338349_4970.c" in 160usr/0sys/408real ms.
Pass ❹: compiled C into "stap_1856e21ea1c246da85ad8c66b4338349_4970.ko" in
2030usr/360sys/10182real ms.
Pass ❺: starting run.
read performed
Pass ❻ (73 ページ): run completed in 10usr/20sys/257real ms.
```

- ❶ /usr/share/systemtap/tapset/ 内にある既存の tap セットライブラリのスクリプトを確認しています。tap セットとは 事前に作成された探査と関数のライブラリ集 (スクリプト) で、SystemTap の スクリプトから利用することができるものです。
- ❷ 含まれているスクリプトに対して確認を行なっています。
- ❸ スクリプトを C 言語のソースコードに変換したあと、インストールされている C コンパイラを実行して、ソースコードをコンパイルしています。それぞれ 生成されたソースコード (*.c) とカーネルモジュール (*.ko) は、~/ .systemtap にある SystemTap キャッシュ内に保持されます。
- ❹ モジュールを読み込み、カーネルに接続することで全ての探査 (イベントとハンドラ) を有効にしています。調査するイベントは仮想ファイルシステム (Virtual File System (VFS)) の読み込み処理です。このイベントはどのような環境でも発生するものなので、正しいハンドラ (read performed と表示するスクリプト) が実行されて正常終了するはずです。
- ❺ SystemTap セッションが完了した後は、探査が無効化されてカーネルモジュールの読み込みが解除されます。

上記のテスト中に何らかのエラーメッセージが表示された場合は、表示されたメッセージを 手がかりにして必要なパッケージを確認し、それらが正しくインストールされているか どうかを確かめてください。場合によっては必要なカーネルを読み込むため、再起動を 行なう必要があるかもしれません。

5.3 スクリプトの文法

SystemTap のスクリプトは下記の 2 つの要素から構成されます:

SystemTap イベント (探査点) (76 ページ)

ハンドラを実行すべきカーネルイベントの名称。イベントとしては特定の 機能 (関数) の呼び出し部分や終了部分に設定することができるほか、タイマー による起動やセッションの起動／終了に設定することができます。

SystemTap ハンドラ (探査体) (77 ページ)

特定のイベントが発生したとき、何をすべきかを記述したスクリプト言語の 羅列。この部分は通常、イベントの状況に応じてデータを抽出し、それらを 内部の変数や結果表示などに保存する動作を行ないます。

イベントとそれに関連するハンドラは、まとめて 探査 と呼ばれます。SystemTap のイベントは 探査ポイント と呼ばれ、そのイベントに対するハンドラは 探査体 と呼ばれます。

SystemTap スクリプト内では、様々な方法で任意の場所にコメントを記述することができます: それぞれ `#`, `/* */`, `//` を使用して記述してください。

5.3.1 調査 (probe) の書式

SystemTap スクリプトでは、複数の調査を記述することができます。調査は下記の形式で記述されなければなりません:

probe イベント {ステートメント}

それぞれの調査には対応するステートメントブロックが存在します。この ステートメントブロックは `{ }` で括られていなければならず、イベントごとにステートメントが存在しなければなりません。

例 5.1 シンプルな SystemTap スクリプト

下記はシンプルな SystemTap スクリプトの例です。

```
probe❶ begin❷
{❸
    printf❹ ("hello world\n")❺
    exit ()❻
}❼
```

- ❶ 調査 (probe) の宣言です。
- ❷ イベント begin (SystemTap セッションの開始時) を指定しています。
- ❸ ハンドラ定義の開始を `{` で示しています。
- ❹ ハンドラ内で最初に使用されている関数 (printf) です。
- ❺ printf 関数で表示すべき文字列が記されています。文字列の最後には改行 (`\n`) が書かれています。
- ❻ ハンドラ内で 2 番目に使用されている関数 (exit()) です。なお SystemTap のスクリプトは、exit() 関数が 呼び出されるまで続けて実行されることに注意してください。それより前の 段階で手動停止させるには、`+ C` を押します。
- ❼ ハンドラ定義の終了を `}` で示しています。

イベント begin ❷ (SystemTap セッションの開始時) が発生すると、`{ }` 内に書かれているハンドラが起動します。この場合は printf 関数 ❹ が呼び出され、hello world という文字列と改行が出力 ❺ されます。出力が終わると、ハンドラは終了 ❸ します。

スクリプト内に複数のステートメントが記述されている場合、SystemTap はこれらの ステートメントを書かれている順番に実行します。ステートメントの間には区切りとなる文字や終了を表わす文字を記述する必要はありません。また、ステートメントの ブロックは他のステートメントのブロック内にネストする (入れ子の形式にする)

こともできます。一般的に SystemTap スクリプトで使用するステートメントブロックは、C プログラミング言語でのそれと同じです。

5.3.2 SystemTap イベント (探査点)

SystemTap では数多くの内蔵イベントをしようすることができます。

一般的なイベントの書式は、ドットで区切って記述する形態です。このような仕組みにより、イベントを分類から絞り込むことができるようになっています。また、それぞれのコンポーネント (イベントのパーツ) では、関数呼び出しのような形式で 文字列 または数値のパラメータを指定することができます。さらに、コンポーネント には * を指定することができます。これは該当する他の探査点に 展開される動作になります。それ以外にも、探査点に ? を付加して、展開に失敗した場合でもエラーとしない、任意指定のものとしてすることができます。なお、探査点に対して ! を付加すると、任意指定かつ十分なもの とすることができます。

SystemTap では探査点に対して複数のイベントを設定することができます。これらは カンマ (,) で区切って指定します。複数のイベントが 単一の探査に対して設定された場合、SystemTap は指定したいずれかのイベントが 発生した際に実行を行いません。

一般的には、イベントは下記のような分類に分けることができます：

- 同期イベント: 任意のプロセスが、カーネルコード内で特定の場所にある命令を実行した場合に発生するイベント。追加の関連データとして、他のイベントに 対する参照ポイント (命令のアドレス) が設定される場合があります。

同期イベントには、たとえば `vfs. ファイル操作` などがあります: 仮想ファイルシステム (Virtual File System (VFS)) に対して、`ファイル操作` で指定するイベントを指定します。たとえば 5.2 項「インストールと設定」(72 ページ) の場合、`read` が VFS で使用されるイベントです。

- 非同期イベント: 特定の命令やコード内での場所に結びつかないものを指します。たとえばカウンタによるものやタイマーによるものなどがあげられます。

非同期イベントとしては、たとえば `begin` などがあります。これは SystemTap セッションの開始を意味するもので、SystemTap スクリプトが 起動するとすぐに実行するものです。それ以外にも `end` (SystemTap セッションの終了時) やタイマーのイベントなどがあります。タイマーイベントは定期的に行うためのハンドラで、たとえば `timer.s(秒)` や `timer.ms(ミリ秒)` などがあります。

情報を収集するためのその他の探査とともに使用した場合、タイマーイベントを利用することで、定期的にデータを表示させることができます。これにより、一定時間ごとにどれだけの変化があったのかを確認することができます。

例 5.2 タイマーイベントによる探査

たとえば、下記の探査では「hello world」というテキストを 4 秒おきに表示します:

```
probe timer.s(4)
{
    printf("hello world\n")
}
```

利用可能なイベントについて、詳しくは `stapprobes` の マニュアルページをお読みください。マニュアルページ内の *See Also* の項には、特定のサブシステムやコンポーネントに対応するイベントについて、他のマニュアルページへのリンクが書かれています。

5.3.3 SystemTap ハンドラ (探査体)

Each SystemTap event is accompanied by a corresponding handler defined for that event, consisting of a statement block.

5.3.3.1 機能

複数の探査で同じステートメント群を使用したい場合は、それらを簡単に 再利用することができます。function の後に名前を 記述することでそれらを関数として定義することができます。関数では 文字列や数値 (値渡し) のパラメータを必要なだけ作成することができます、単一の文字列や数値を返却することができます。

```
function 関数名(パラメータ) {ステートメント}
probe イベント {関数名(パラメータ)}
```

上記の例において、*関数名* 内にあるステートメントは、*イベント* で指定した探査が発生したタイミングで 実行されます。*パラメータ* は任意指定の値で、関数に渡すパラメータを指定します。

関数はスクリプト内の任意の場所に定義することができます。

例5.1「シンプルな SystemTap スクリプト」(75 ページ) で既に紹介しており、とてもよく使用される関数のうちの 1 つが `printf` 関数です。これは書式を

指定してデータを出力する関数で、書式指定文字列を指定することで、どのような形態で出力するのかが設定できます。書式指定文字列は引用符で括って指定し、その中で % 文字で始まる詳細な書式指定を行ないます。

書式指定文字列にどのように指定するのかは、表示対象のパラメータによって異なります。書式指定文字列には複数の書式指定を設定することができ、それぞれパラメータ 1 つに対応します。複数のパラメータはカンマで区切ります。

例 5.3 書式指定付きの printf 関数

```
printf ("❶%s❷(%d❸) open❹n", execname(), pid())
```

- ❶ 書式指定文字列の始まり (") を指定しています。
- ❷ 文字列の書式指定です。
- ❸ 整数の書式指定です。
- ❹ 書式指定文字列の終わり (") を指定しています。

上記の例では、現在の実行名 (execname()) を文字列で、プロセス ID (pid()) を整数で括弧内に表示したあと、スペースで区切って open という文字列を表示して改行します：

```
[...]
vmware-guestd(2206) open
hald(2360) open
[...]
```

例5.3「書式指定付きの printf 関数」(78 ページ) で使用している execname() と pid() の 2 つの関数以外にも、printf では様々な関数を使用することができます。

SystemTap のスクリプトでもっともよく使用する関数は下記のとおりです：

tid()
現在のスレッドの ID

pid()
現在のスレッドのプロセス ID

uid()
現在のユーザ ID

cpu()
現在の CPU 番号

execname()

現在のプロセスの名称

gettimeofday_s()

UNIX エポック (1970 年 1 月 1 日) からの経過秒数

ctime()

時刻を文字列に変換したもの

pp()

現在処理している探査点の説明文字列

thread_indent()

出力結果の体裁を整えるのに便利な関数です。この関数では各スレッド (tid()) に対して、インデントカウンタを保持する 動作を行ないます。この関数はインデント差分値と呼ばれる、スレッドの インデントカウンタを増減させるための 1 つのパラメータを指定します。関数の返り値は、インデントに必要な適切な数のスペースと、いくつかの 汎用的な追跡データを含む文字列です。汎用的な追跡データとしては、タイムスタンプ (スレッドに対する最初のインデント処理からの経過 マイクロ秒数) のほか、プロセス名とスレッド ID 自身が含まれます。これにより、呼び出された関数名や呼び出し元、およびスレッドの経過 時間を観測することができるようになっています。

関数の開始から終了までの間には通常、別の関数の呼び出しによる開始や 終了が含まれることから、何も行なわない状態ではそれらを識別するのが 難しくなってしまいます。このインデントカウンタでは、*別の* 関数が呼び出された場合に値を増やし、そこから帰ってきた段階で値を減らすことができるので、関数の呼び出しが 起こるたびに右にずらす表示を行ない、関数の開始と終了を見分けやすく する効果があります。thread_indent() を使用する SystemTap スクリプトの例については、<http://sourceware.org/systemtap/tutorial/Tracing.html#fig:socket-trace> にある *SystemTap Tutorial* をお読みください。

利用可能な SystemTap 関数について、詳しくは stapfuncs のマニュアルページをお読みください。

5.3.3.2 その他の基本的な構造

関数とは別に、SystemTap のハンドラではその他の一般的な構造を利用することができます。たとえば変数や条件分岐 (if/ else)、ループ (while, for)、配列やコマンドラインパラメータなどがあります。

変数

変数はスクリプト内の任意の場所で定義することができます。変数を定義するには、単純に名前を指定して関数の値や数式を代入してください:

```
foo = gettimeofday( )
```

すると、変数を数式などで使用できるようになります。変数に代入した値の種類によって、SystemTap は自動的にデータ型を推測 (文字列または 数値) します。何らかの矛盾が発生すると、それらはエラーとして報告されます。上記の例では foo は自動的に数値として解釈され、printf() で整数の書式を指定 (%d) することによって表示できるようになります。

ただし、既定の設定では変数は、それが使用されている探査内ローカルの存在です。つまり、変数はそれぞれのハンドラ内で初期化されて使用され、そして廃棄されます。変数を探査の間で共有するには、スクリプト内でその変数をグローバル変数として宣言する必要があります。これを行なうには、探査の外側で global キーワードを使用します:

例 5.4 グローバル変数の使用

```
global count_jiffies, count_ms
probe timer.jiffies(100) { count_jiffies ++ }
probe timer.ms(100) { count_ms ++ }
probe timer.ms(12345)
{
    hz=(1000*count_jiffies) / count_ms
    printf ("jiffies:ms ratio %d:%d => CONFIG_HZ=%d\n",
           count_jiffies, count_ms, hz)
    exit ()
}
```

上記のサンプルスクリプトでは、カーネルの CONFIG_HZ の設定をタイマーで計算するもので、それぞれカーネルの時分割値 (jiffies) とミリ秒を数えています (jiffy/jiffies はシステムのタイマー割り込みの 1 カウント分を意味する値です。ハードウェアプラットフォーム側で決める割り込み周期に依存するため、プラットフォームごとに異なる値になります)。global ステートメントを使用することで count_jiffies と count_ms の変数をグローバル変数としていて、timer.ms(12345) の中でも使用できるようになっています。なお、++ は指定した変数の値に 1 を加える、という意味です。

条件分岐

SystemTap スクリプト内で使用できる条件分岐には、いろいろなものがあります。よく使用されるものを下記に示します:

If/Else ステートメント

下記のような書式で記述します:

```
if (条件)①ステートメント_1②  
    else③ステートメント_2④
```

if ステートメントは整数値を 0 と比較します。条件演算子 ① が 0 以外の場合、最初のステートメント ② が実行されます。0 の場合は 2 番目のステートメント ④ が実行されます。else の指定 (③ および ④) は任意で、指定しなくてもかまいません。また、② と ④ は、いずれもステートメントブロックを記述することができます。

While ループ

下記のような書式で記述します:

```
while (条件)①ステートメント②
```

条件 で指定した値が 0 以外である限り、② で指定したステートメントを繰り返し実行します。なお、② はステートメントブロックを記述することもできます。条件 の値は何回かの繰り返しの後に 0 にならなければなりません。

For ループ

基本的には while に対するショートカット機能で、下記のような書式で記述します:

```
for (初期化①; 条件②; 増加表現③) statement
```

① で指定した表現は繰り返し処理の初期化の際に使用するもので、処理が 実際に行なわれる前に実行される項目です。実行は ② で指定した条件が満たされなくなるまで繰り返し行なわれます (この表現は 各繰り返し処理の冒頭で行なわれます)。また、③ で指定した表現は、ループカウンタの加算に使用します。この表現は各 繰り返し処理の終わりで行なわれます。

条件演算子

条件演算子としては下記のものを使用することができます:

==: 左辺と右辺が等しい

!=: 左辺と右辺が等しくない

>=: 左辺のほうが右辺より大きい、等しい

<=: 左辺のほうが右辺より小さい、等しい

5.4 スクリプト例

systemtap-docs パッケージをインストールしていれば、`/usr/share/doc/packages/systemtap/examples` 内に各種の便利な SystemTap スクリプト例が配置されます。

この章では、`/usr/share/doc/packages/systemtap/examples/network/tcp_connections.stp` にあるサンプルのスクリプトについて、より詳しく説明します。

例 5.5 *tcp_connections.stp* を利用した着信 TCP の監視

```
#!/usr/bin/env stap

probe begin {
    printf("%6s %16s %6s %6s %16s\n",
           "UID", "CMD", "PID", "PORT", "IP_SOURCE")
}

probe kernel.function("tcp_accept").return?,
       kernel.function("inet_csk_accept").return? {
    sock = $return
    if (sock != 0)
        printf("%6d %16s %6d %6d %16s\n", uid(), execname(), pid(),
           inet_get_local_port(sock), inet_get_ip_source(sock))
}
```

この SystemTap スクリプトは着信側の TCP 接続をリアルタイムに監視し、許可されていない接続や望まない接続が発生しているかどうかを確認します。それぞれ TCP の接続がコンピュータ側に受け入れられると、下記のような情報が表示されます：

- ユーザ ID (UID)
- 接続を受け入れているコマンド (CMD)
- コマンドのプロセス ID (PID)
- 接続が使用しているポート (PORT)
- TCP の接続元 IP アドレス (IP_SOURCE)

スクリプトを実行するには、下記のように入力します：

```
stap /usr/share/doc/packages/systemtap/examples/network/tcp_connections.stp
```


あとはスクリプトの実行結果が出力されます。スクリプトを停止させるには、+ C を押します。

5.5 さらなる情報

この章では、SystemTap についての概要を説明してきました。SystemTap についてより詳しい情報を得るには、下記のリンクを参照してください:

<http://sourceware.org/systemtap/>

SystemTap プロジェクトの Web ページです。

<http://sourceware.org/systemtap/wiki/>

SystemTap に関する便利な情報が数多く記載されています。レビューや他のツール との比較、よくある質問 (FAQ) やヒントなど、ユーザ向けおよび開発者向けのドキュメンテーションがあります。また、SystemTap のスクリプト集も掲載されているほか、サンプルや使用例、最新の議論やストーリーなどもあります。

<http://sourceware.org/systemtap/documentation.html>

SystemTap Tutorial (SystemTap チュートリアル), *SystemTap Beginner's Guide* (SystemTap 初心者向けガイド), *Tapset Developer's Guide* (SystemTap 開発者向けガイド), *SystemTap Language Reference* (SystemTap 言語リファレンス) が、それぞれ PDF と HTML の形式で公開されています。また、これに関連した マニュアルページも公開されています。

SystemTap の言語リファレンスや SystemTap のチュートリアルについては、パッケージを インストールしたシステムであれば `/usr/share/doc/packages/systemtap` 内にも存在します。また、SystemTap のサンプルスクリプトは上記の `example` サブディレクトリにあります。

カーネル探査

カーネルの探査とは、Linux カーネルのデバッグや性能に関する情報を集めるための、一連のツール集のことを指します。開発者やシステム管理者は通常、これらを使用してカーネルのデバッグを行ったり、システム性能のボトルネックを発見しようとしたりします。ここから報告されたデータは、システムの性能を改善する目的で使われます。

探査はカーネルのルーチン (一連の動作) の中に組み込むことができ、特定のポイントに到達した時に実行されるハンドラを設定することができます。カーネル探査の主な利点としては、カーネルの再構築を行ったりする必要がなくなるほか、探査項目に変更を加える際でも再起動を必要としない点が挙げられます。

カーネル探査を使用するには、通常は特定のカーネルモジュールを作成したり取得したりする必要があります。このようなモジュールには *init* (初期化) と *exit* (終了) の各関数が含まれています。init 関数 (`register_kprobe()` など) では 1 つまたは複数の探査を登録し、exit 関数ではこれらの登録を解除します。登録関数では、探査を挿入する場所を指定するほか、探査に該当した場合に、どのハンドラを呼び出すのかを指定します。一括で複数の探査を登録または登録解除するには、`register_<探査の種類>probes()` や `unregister_<探査の種類>probes()` を使用することができます。

デバッグメッセージと状態を表わすメッセージは、一般に `printk` カーネルルーチンで実施します。`printk` はユーザスペースで言うところの `printf` と同じルーチンです。`printk` について、詳しくは [Logging kernel messages \[http://www.win.tue.nl/~aeb/linux/lk/lk-2.html#ss2.8\]](http://www.win.tue.nl/~aeb/linux/lk/lk-2.html#ss2.8) (英語) をお読みください。通常であれば、これらのメッセージは `/var/log/messages` や `/var/log/syslog` を参

照することで、確認することができます。詳しくは 第4章 システムログファイルの分析と管理 (59 ページ) をお読みください。

6.1 サポートされるアーキテクチャ

カーネル探査は下記のアーキテクチャで *完全に* 実装されています:

- i386
- x86_64 (AMD-64, EM64T)
- ppc64
- arm
- ppc

カーネル探査は下記のアーキテクチャで *部分的に* 実装されています:

- ia64 (slot1 命令に対する探査には対応していません)
- sparc64 (return 探査が実装されていません)

6.2 カーネル探査の種類

カーネル探査には 3 つの種類があります: *Kprobes*, *Jprobes*, *Kretprobes*。
kretprobes とは *return probe* と呼ばれます。これら 3 種類の探査のソースコード例は、`/usr/src/linux/samples/kprobes/` 内 (kernel-source パッケージ) に存在しています。

6.2.1 Kprobe

Kprobe は Linux カーネル内の任意の命令に対して設定することができるものです。Kprobe を特定の命令に対して設定すると、その命令が書かれている 最初の場所に対してブレイクポイントが設定されます。プロセッサが ブレイクポイントに到達すると、プロセッサのレジスタ情報が保存されたあと 処理が Kprobe に渡されま

す。最初に *プリハンドラ* (事前のハンドラ) が実行され、そのあと対象の命令が一つずつ実行され、最後に *ポストハンドラ* (事後のハンドラ) が実行されます。これら一連の流れが終わると、制御は Kprobe で保存しておいたアドレスに戻ります。

6.2.2 Jprobe

Jprobe は Kprobe の仕組みを通して実装されているものです。これは関数の 開始点 (エントリポイント) に対して設定されるもので、対象の関数に対する パラメータについて、直接的なアクセスを行なうことができますようになっています。このハンドラルーチンは、関数と同じパラメータ一覧および返り値を設定しなければなりません。また、jprobe_return() を呼び出すことで 終了しなければなりません。

Jprobe での探査に引っかけると、プロセッサのレジスタ情報が保存された あと、命令のポインタが Jprobe のハンドラルーチン側に向けられます。その後、制御は関数が呼び出されたときと同じレジスタ状態で、ハンドラを実行します。最後にハンドラは jprobe_return() 関数を呼び出し、元の関数に戻ります。

一般的には 1 つの関数に複数の探査を挿入することが可能ですが、Jprobe では 1 つの関数あたり 1 つのインスタンスしか設定できません。

6.2.3 Return probe

Return probeについても同様に、KProbe の仕組みを通して実装されています。register_kretprobe() 関数が呼び出されると、KProbe が指定の関数の開始点に設定されます。探査に引っかけると、カーネルの探査機構が対象の関数の戻りアドレスを保存し、あらかじめ設定しておいたハンドラを呼び出します。これら一連の流れが終わると、制御は Return probe で保存しておいたアドレスに戻ります。

なお register_kretprobe() を呼び出す前に、maxactive パラメータを設定する必要があります。これは同時にどれだけのインスタンス数を探査するかを指定します。この値が小さすぎると、探査を見逃してしまう場合があります。

6.3 カーネルの探査 API

Kprobe のプログラミングインターフェイスは複数の関数から構成されていて、これらは 使用する全てのカーネル探査に対して、登録や登録解除を行なうことができます。ほか、ハンドラを関連づけたりすることができます。これらの関数について、詳しい説

明や それらのパラメータについては、6.5項「さらなる情報」(89 ページ) に示す情報を参照してください。

`register_kprobe()`

指定したアドレスにブレイクポイントを設定します。ブレイクポイントに到達すると、`pre_handler` と `post_handler` がそれぞれ呼び出されます。

`register_jprobe()`

指定したアドレスにブレイクポイントを設定します。アドレスは探査対象の関数に対して最初の命令位置でなければなりません。ブレイクポイントに到達すると、指定した ハンドラが実行されます。ハンドラは探査対象の関数と同じパラメータリスト、かつ返却型であるべきものです。

`register_kretprobe()`

指定した関数に `return probe` を設定します。探査対象の関数が呼び出されると、指定したハンドラが実行されます。この関数は成功時に 0 を、失敗時には負の数を 返却します。

`unregister_kprobe()`, `unregister_jprobe()`, `unregister_kretprobe()`

それぞれ指定した探査を削除します。登録後であればいつでも呼び出すことが可能です。

`register_kprobes()`, `register_jprobes()`, `register_kretprobes()`

それぞれ指定した配列内にある複数の探査を設定します。

`unregister_kprobes()`, `unregister_jprobes()`, `unregister_kretprobes()`

それぞれ指定した配列内にある複数の探査を削除します。

`disable_kprobe()`, `disable_jprobe()`, `disable_kretprobe()`

指定した探査を一時的に無効化します。

`enable_kprobe()`, `enable_jprobe()`, `enable_kretprobe()`

一時的に無効化されていた探査を有効にします。

6.4 Debugfs インターフェイス

最近の Linux カーネルであれば、カーネル探査の命令はカーネル内にある `debugfs` インターフェイスを使用します。これにより、全ての登録済み探査を一覧表示できる ほか、全ての探査に対して有効や無効を切り替えることができます。

6.4.1 登録済みのカーネル探査を一覧表示する方法

現在登録されている全ての Kprobe を表示するには、`/sys/kernel/debug/kprobes/list` ファイルをお読みください。

```
saturn.example.com:~ # cat /sys/kernel/debug/kprobes/list
c015d71a k  vfs_read+0x0  [DISABLED]
c011a316 j  do_fork+0x0
c03dedc5 r  tcp_v4_rcv+0x0
```

最初の列には探査が設定されたカーネル内のアドレス情報が表示されます。2 番目の列には探査の種類が書かれています。それぞれ `k` が Kprobe、`j` が Jprobe、`r` が Return probe を示しています。3 番目の列にはカーネル内のシンボルとオフセット値、および探査の状態に関する情報が表示されます。その時点で既に有効ではない仮想 アドレスに探査が位置している場合は、`[GONE]` という マークが併記されます。また、一時的に無効化された探査の場合は、`[DISABLED]` というマークが併記されます。

6.4.2 全てのカーネル探査を有効または無効にする方法

`/sys/kernel/debug/kprobes/enabled` ファイルには、登録済みの全てのカーネル探査について、有効または無効に切り替えることのできる 機能が用意されています。全てのカーネル探査を無効にするには、`root` で 下記のように実行します：

```
echo "0" > /sys/kernel/debug/kprobes/enabled
```

全てのカーネル探査を有効に戻すには、同様に `root` で下記のように実行します：

```
echo "1" > /sys/kernel/debug/kprobes/enabled
```

なお、この方法では探査の状態を変更することはできません。特定の探査が一時的に 無効化されている場合、上記のコマンドを実行しても `[DISABLED]` の状態のままになります。

6.5 さらなる情報

カーネル探査についてより詳しく知るには、下記の情報源をご利用ください：

- カーネル探査に関して、完全かつ技術的な情報を参照したい場合は、`/usr/src/linux/Documentation/kprobes.txt` ファイル (`kernel-source` パッケージ内) をお読みください。
- 3 種類の探査についてのサンプル (および関連する `Makefile`) をお読みになりたい場合は、`/usr/src/linux/samples/kprobes/` ディレクトリ (`kernel-source` パッケージ内) をご覧ください。
- Linux カーネルモジュールや `printk` カーネルルーチンについて詳しい情報を得るには、`The Linux Kernel Module Programming Guide` [<http://tldp.org/LDP/lkmpg/2.6/html/lkmpg.html>] (英語) をお読みください。
- カーネル探査を実際に使用するにあたって、現実的ではありますが少し古い情報を得たい場合は、`Kernel debugging with Kprobes` [<http://www.ibm.com/developerworks/library/l-kprobes.html>] (英語) をお読みください。

perfmon2—ハードウェアベースの性能監視

perfmon2 はプロセッサ性能監視ユニット (PMU) にアクセスするための標準化・汎用 インターフェイスです。各種の PMU モデルやアーキテクチャに対して、可搬性のある アクセス機能を提供します。これによりシステム全体だけでなくスレッド単位での 監視やカウント、およびサンプリングを行なうことができるようになっています。

7.1 概念

perfmon の概要については下記をお読みください。

7.1.1 perfmon2 の構造

性能監視とは、「どのようにアプリケーションやシステムが動作しているのかについて、情報を収集する行為」のことを指します。これらの情報は、動作しているコードから取得することができるほか、CPU やチップセットなどからも取得することができます。

新しいプロセッサであれば、性能監視ユニット (PMU) と呼ばれるものが搭載 されています。PMU の設計と機能は CPU ごとに異なる仕様になっていて、たとえばレジスタの数やカウンタの数、機能数については、CPU 側の実装によって異なります。

perfmon インターフェイスは汎用的で柔軟性や拡張性に富んだ設計になっています。そのため、プログラム (スレッド) だけでなくシステム全体のレベル でも監視を行なうことができるようになっていて、いずれの場合でもプロファイル 情報をカウント

したりサンプリングしたりすることができるようになっています。この統一されたインターフェイスにより、可搬性のあるツールを作成することができるようになっています。まずは 図7.1「perfmon2 のアーキテクチャ」(92 ページ) で概要をお読みください。

図 7.1 perfmon2 のアーキテクチャ

各 PMU は複数のレジスタから構成されています。それぞれ性能モニタ設定 (PMC) と性能モニタデータ (PMD) と呼ばれます。書き込みは PMC 側にだけ 行なうことができますが、読み込みは両方のレジスタで行なうことができます。これらのレジスタには、設定情報とデータが保存されます。

7.1.2 サンプリングとカウント

perfmon2 は 2 種類のモードに対応します。それぞれサンプリングとカウント と呼ばれるものです。

サンプリングとは通常、一定の時間間隔 (時間ベース) や 一定回数のイベント発生間隔 (イベントベース) で処理されるモードです。perfmon の場合、時間ベースのサンプリングは、時間で発生するイベント (たとえば `unhalted_reference_cycles` など) を利用して間接的に 実現しています。

一方、カウント はイベントの発生回数を元に処理する モードです。

いずれの情報であっても、それらの情報は サンプル 内に保管されます。このサンプルには、たとえばどのスレッドで発生したものかや、命令ポインタの情報などが含まれています。

下記の例では CPU_OP_CYCLES イベントの発生回数を数え、100000 回発生するごとにサンプルを採取します:

```
pfmon --no-cmd-output -e CPU_OP_CYCLES_ALL /bin/ls
1306604 CPU_OP_CYCLES_ALL
```

また、下記のコマンドは指定した機能の実行回数と、全体サイクル内での 割合を表示します:

```
pfmon --no-cmd-output --short-smpl-periods=100000 -e CPU_OP_CYCLES_ALL /bin/ls
# results for [28119:28119<-[28102]] (/bin/ls)
# total samples          : 12
# total buffer overflows : 0
#
#                          event00
```

```
# counts %self %cum code addr
      1 8.33% 8.33% 0x2000000000007180
      1 8.33% 16.67% 0x20000000000195a0
      1 8.33% 25.00% 0x2000000000019260
      1 8.33% 33.33% 0x2000000000014e60
      1 8.33% 41.67% 0x200000000001f38c0
      1 8.33% 50.00% 0x200000000001ea481
      1 8.33% 58.33% 0x2000000000020b260
      1 8.33% 66.67% 0x20000000000203490
      1 8.33% 75.00% 0x20000000000203360
      1 8.33% 83.33% 0x20000000000203440
      1 8.33% 91.67% 0x40000000000002690
      1 8.33% 100.00% 0x200000000001cfd1
```

7.2 インストール

perfmon2 を使用するには、まず下記の要件が満たされていることをご確認ください:

SUSE Linux Enterprise 11

対応するアーキテクチャは、それぞれ IA64, x86_64 です。perf (Linux 向け パフォーマンス カウンタ) パッケージの場合は、x86, PPC64 の各アーキテクチャに対応しています。

SUSE Linux Enterprise 11 SP1

対応するアーキテクチャは IA64 のみです。

SUSE Linux Enterprise11 上の pfmon の場合、それぞれ下記のプロセッサに 対応しています (/usr/share/doc/packages/pfmon/README からの抜粋):

表 7.1 サポート対象のプロセッサ

型式	プロセッサ
Intel IA-64	Itanium (Merced), Itanium 2 (McKinley, Madison, Deerfield), Itanium 2 9000/9100 (Montecito, Montvale) など
AMD X86	Opteron (K8, fam 10h)
Intel X86	Intel P6 (Pentium II, Pentium Pro, Pentium III, Pentium

型式	プロセッサ
	M); Yonah (Core Duo, Core Solo); Netburst (Pentium 4, Xeon); Core (Merom, Penryn, Dunnington) Core 2 または Quad; Atom; Nehalem; architectural perfmon v1, v2, v3

お使いのアーキテクチャごとに必要なパッケージが異なります。必要なパッケージをインストールしてください:

表 7.2 必要なパッケージ

アーキテクチャ	パッケージ
ia64	pfmon

7.3 perfmon の使用

perfmon を使用する場合、pfmon と呼ばれるコマンドライン ツールを利用してすべての情報収集を行ないます。

注記: perfmon と OProfile セッションの同時使用について

x86 アーキテクチャの場合、perfmon のセッションと OProfile のセッションを同時に起動することはできません。一方をお使いの場合は、他方を停止させてください。

7.3.1 イベント情報の取得

利用可能なイベントの一覧を取得するには、pfmon コマンドに `-l` を付与して実行してください。なお、この一覧はホスト側の PMU によってそれぞれ異なることに注意してください:

```
pfmon -l
ALAT_CAPACITY_MISS_ALL
ALAT_CAPACITY_MISS_FP
ALAT_CAPACITY_MISS_INT
BACK_END_BUBBLE_ALL
```

```

BACK_END_BUBBLE_FE
BACK_END_BUBBLE_L1D_FPU_RSE
...
CPU_CPL_CHANGES_ALL
CPU_CPL_CHANGES_LVL0
CPU_CPL_CHANGES_LVL1
CPU_CPL_CHANGES_LVL2
CPU_CPL_CHANGES_LVL3
CPU_OP_CYCLES_ALL
CPU_OP_CYCLES_QUAL
CPU_OP_CYCLES_HALTED
DATA_DEBUG_REGISTER_FAULT
DATA_DEBUG_REGISTER_MATCHES
DATA_EAR_ALAT
...

```

各項目について説明を読みたい場合は、`-i` に続けてイベント名 を指定します:

```

pfmon -i CPU_OP_CYCLES_ALL
Name      : CPU_OP_CYCLES_ALL
Code      : 0x12
Counters  : [ 4 5 6 7 8 9 10 11 12 13 14 15 ]
Desc      : CPU Operating Cycles -- All CPU cycles counted
Umask     : 0x0
EAR       : None
ETB       : No
MaxIncr   : 1 (Threshold 0)
Qual      : None
Type      : Causal
Set       : None

```

7.3.2 システム全体に対するセッションの有効化

`--system-wide` オプションを使用すると、特定の CPU (単体または複数) 上で実行されるすべてのプロセスを監視することができます。この場合も、`root` で実行する必要はありません。既定ではユーザレベルが 全イベントに対して有効になっているためです (`-u` オプション)。

また、同じ CPU を監視しない限り、複数のシステム全体セッションを同時に実行することもできます。ただし、スレッドごとのセッションとシステム全体のセッションは 同時に実行できません。

下記の例では、Itanium IA64 Montecito プロセッサを利用して採取した例です。システム全体のセッションを実行するには、下記の手順を行ないます:

1 お使いの CPU セットを検出します:

```

pfmon -v --system-wide
...

```

selected CPUs (2 CPU in set, 2 CPUs online): CPU0 CPU1

- 2 セッションに制約を設定します。下記の一覧では、以降の例で使用するオプションについて説明を記述しています (詳しくはマニュアルページをお読みください):

`-e/--events`

選択したイベントのみをプロファイルします。一覧を取得するには、7.3.1項「イベント情報の取得」(94 ページ)をお読みください。

`--cpu-list`

監視対象のプロセッサ一覧を指定します。このオプションを指定しない場合、利用可能なすべてのプロセッサを監視します。

`-t/--session-timeout`

監視セッションの実行時間を秒単位で指定します。

プロファイルセッションを開始するには、上記のいずれかの方法をご利用ください。

- 既定のイベントを使用します:

```
pfmon --cpu-list=0-2 --system-wide -k -e CPU_OP_CYCLES_ALL,IA64_INST_RETIRED
<press ENTER to stop session>
CPU0          7670609 CPU_OP_CYCLES_ALL
CPU0          4380453 IA64_INST_RETIRED
CPU1          7061159 CPU_OP_CYCLES_ALL
CPU1          4143020 IA64_INST_RETIRED
CPU2          7194110 CPU_OP_CYCLES_ALL
CPU2          4168239 IA64_INST_RETIRED
```

- 秒単位で実行時間を指定します:

```
pfmon --cpu-list=0-2 --system-wide --session-timeout=10 -k -e
CPU_OP_CYCLES_ALL,IA64_INST_RETIRED
<session to end in 10 seconds>
CPU0          69263547 CPU_OP_CYCLES_ALL
CPU0          38682141 IA64_INST_RETIRED
CPU1          87189093 CPU_OP_CYCLES_ALL
CPU1          54684852 IA64_INST_RETIRED
CPU2          64441287 CPU_OP_CYCLES_ALL
CPU2          37883915 IA64_INST_RETIRED
```

- コマンドを実行します。セッションはプログラムを起動することで自動的に 開始され、プログラムが終了すると自動的に停止されます:

```
pfmon --cpu-list=0-1 --system-wide -u -e CPU_OP_CYCLES_ALL,IA64_INST_RETIRED -- ls -
l /dev/null
```

```
crw-rw-rw- 1 root root 1, 3 27. Mär 03:30 /dev/null
CPU0          38925 CPU_OP_CYCLES_ALL
CPU0          7510 IA64_INST_RETIRED
CPU1          9825 CPU_OP_CYCLES_ALL
CPU1         1676 IA64_INST_RETIRED
```

3 キーを押すとセッションを停止することができます:

4 合計値を取得したい場合は、対象となるコマンドの後に `-aggr` オプションを指定します:

```
pfmon --cpu-list=0-1 --system-wide -u -e CPU_OP_CYCLES_ALL,IA64_INST_RETIRED --aggr
<press ENTER to stop session>
```

```
52655 CPU_OP_CYCLES_ALL
53164 IA64_INST_RETIRED
```

7.3.3 動作中の処理の監視

`perfmon` は動作中のスレッドを監視することもできます。これは、起動に時間のかかるシステムデーモンやプログラムを監視する場合に便利な機能です。この場合は、まず監視対象のプロセス ID を判断します:

```
ps ax | grep foo
10027 pts/1    R      2:23   foo
```

あとは見つかったプロセス ID を、`pfmon` コマンドの `--attach-task` オプションで指定します:

```
pfmon --attach-task=10027
3682190 CPU_OP_CYCLES_ALL
```

7.4 debugfs からの統計情報取得

`perfmon` は、デバッグ用のインターフェイスで公開されている統計情報を収集することもできます。この方法では、多くが合計値としてのカウントや 間隔として採取されます。

データへのアクセスは、`root` で `/sys/kernel/debug` 以下にマウントされた、デバッグファイルシステムを通じて行ないます。

データは `/sys/kernel/debug/perfmon/` 内に CPU 単位で 存在しています。それぞれの CPU には、ASCII ファイル形式でアクセス可能な 複数の測定値が含まれています。下記の情報は `/usr/src/linux/Documentation/perfmon2-debugfs.txt` からの抜粋です:

表 7.3 `/sys/kernel/debug/perfmon/cpu*/` 内にある読み取り専用ファイル

ファイル	説明
<code>ctxswin_count</code>	PMU コンテキスト切り替えの突入回数
<code>ctxswin_ns</code>	PMU コンテキスト切り替え突入ルーチン内で消費された時間 (ナノ秒単位) Average cost of the PMU context switch in = $\text{ctxswin_ns} / \text{ctxswin_count}$
<code>ctxswout_count</code>	PMU コンテキスト切り替えの解放回数
<code>ctxswout_ns</code>	PMU コンテキスト切り替え解放ルーチン内で消費された時間 (ナノ秒単位) Average cost of the PMU context switch out = $\text{ctxswout_ns} / \text{ctxswout_count}$
<code>fmt_handler_calls</code>	PMU の割り込みを処理する、サンプル採取書式化ルーチン (一般的にはサンプルを記録するルーチン) の呼び出し回数
<code>fmt_handler_ns</code>	PMU サンプル採取書式化ルーチン内で消費された時間 (ナノ秒単位) Average time spent in this routine = $\text{fmt_handler_ns} / \text{fmt_handler_calls}$
<code>handle_timeout_count</code>	<code>pfm_handle_timeout()</code> ルーチン (時間ベースのモードで使用されます) の呼び出し回数
<code>handle_work_count</code>	<code>pfm_handle_work()</code> ルーチンの呼び出し回数

ファイル	説明
ovl_intr_all_count	カーネルが受信した PMU 割り込みの回数
ovfl_intr_nmi_count	perfmon からカーネルが受信した、マスク不可能な割り込み (NMI) の回数 (X86 ハードウェアのみ)
ovfl_intr_ns	perfmon2 PMU 割り込み処理ルーチン内で消費された時間 (ナノ秒単位) Average time to handle one PMU interrupt = $\text{ovfl_intr_ns} / \text{ovfl_intr_all_count}$
ovfl_intr_regular_count	perfmon の割り込み処理ルーチンが処理を行なった PMU 割り込みの回数
ovfl_intr_replay_count	コンテキスト切り替えの突入やイベントセットの切り替えなどで再実行された PMU 割り込みの回数
perfom_intr_spurious_count, ovfl_intr_spurious_count	有効なコンテキスト情報が存在しなかったために捨てられた、PMU 割り込みの回数
pfm_restart_count	pfm_restart() が呼び出された回数
reset_pmds_count	pfm_reset_pmds() が呼び出された回数
set_switch_count	イベントセットの切り替え回数
set_switch_ns	セット切り替えルーチン内で消費された時間 (ナノ秒単位) Average cost of switching sets =

ファイル	説明
	set_switch_ns / set_switch_count

perfmon の実行前後でこれらの情報を比較するという方法もあります。たとえば下記のようにしてデータを採取します:

```
for i in /sys/kernel/debug/perfmon/cpu0/*; do
    echo "$i:"; cat $i
done >> pfmon-before.txt
```

次に、特定の CPU に限定して性能監視を行ないます:

```
pfmon --cpu-list=0 ...
```

再度データを採取します:

```
for i in /sys/kernel/debug/perfmon/cpu0/*; do
    echo "$i:"; cat $i
done >> pfmon-after.txt
```

採取した 2 つの値を比較します:

```
diff -u pfmon-before.txt pfmon-after.txt
```

7.5 さらなる情報

この章では概要のみを説明してきました。より詳しい情報については、それぞれ 下記のリンク先をお読みください:

<http://perfmon2.sourceforge.net/>
プロジェクトの Web ページです。

http://www.iop.org/EJ/article/1742-6596/119/4/042017/jpconf8_119_042017.pdf
PDF 形式での概要説明です。

第8章 *OProfile*—システム全体に対するプロファイラ (101 ページ)
その他の性能最適化については、こちらの章をお読みください。

OProfile—システム全体に対する プロファイラ

OProfile は動的なプログラム分析を行なうためのプロファイラです。実行中の プログラムに対してその動作を調査し、情報を収集することができます。収集した 情報は、さらなる最適化を行なうためのヒントとして閲覧することができます。

OProfile を使用するにあたっては、ラッパーライブラリを使用するように修正 したり、コンパイルし直したりする必要はありません。カーネルに対する修正も 不要です。アプリケーションをプロファイルする場合、プロファイル作業の負荷と 採取間隔にもよりますが、通常の使用範囲であれば、ごく少量のオーバーヘッドが あるだけです。

8.1 考え方の概要

OProfile は、カーネルドライバとデータを収集するためのデーモンから構成されています。また、Intel, AMD などのプロセッサが提供する、パフォーマンス カウンタと呼ばれるハードウェアを使用して行ないます。OProfile はカーネルや カーネルモジュール、カーネルの割り込みハンドラやシステムの共有ライブラリ、その他のアプリケーションなど、すべてのコードをプロファイルすることができます。

新しいプロセッサであれば、パフォーマンスカウンタとしてハードウェア経由の プロファイルを行なうことができます。プロセッサの種類にもよりますが、多くの カウンタが用意されていて、それぞれカウント対象のイベントをプログラムすることが できます。またそれぞれのカウンタでは、サンプルを採取する頻度を設定することが できます。小さい値ほどより頻繁に採取する意味になります。

最後に事後の処理では、すべての情報を収集したあと命令アドレスを各関数名に置き換える作業を行ないます。

8.2 インストールと要件

OProfile を使用するには `oprofile` パッケージをインストールします。OProfile はそれぞれ IA-64, AMD64, s390, PPC64 の各プロセッサ上で動作します。

なお、プロファイル対象のアプリケーションに対して提供されている、`*-debuginfo` パッケージをインストールしておくことを お勧めします。カーネルをプロファイルしたい場合も同様に、`debuginfo` パッケージをインストールしてください。

8.3 利用可能な OProfile ユーティリティ

OProfile は複数のユーティリティから構成され、それぞれプロセスのプロファイル処理を行なうものと、そのプロファイルデータを処理するものに分かれています。下記では、本章内で使用する各プログラムについて、概要を説明しています：

`opannotate`

ソースコードやアセンブリに対して注釈を設定し、プロファイル情報に埋め込むコマンドです。

`opcontrol`

プロファイルセッションを制御 (開始や停止) したり、プロファイルデータを ダンプしたり、パラメータを設定したりします。

`ophelp`

利用可能なイベントの一覧と、それらに対する簡単な説明を表示します。

`opimport`

サンプルのデータベースファイルを、対外的なバイナリ形式から ネイティブな形式に変換します。

`opreport`

プロファイルデータからレポートを生成します。

8.4 OProfile の使用

OProfile はカーネルとアプリケーションの両方をプロファイルできます。カーネルをプロファイルする場合は、OProfile に対して `vmlinux*` の場所を指定します。`--vmlinux` オプションを指定して、`vmlinux*` の場所 (通常であれば `/boot` 内) を指定してください。カーネルモジュールをプロファイルする必要がある場合は、OProfile に対して特に何も指定する必要はありません。ただし、あらかじめ <http://oprofile.sourceforge.net/doc/kernel-profiling.html> (英語) を読んでおいてください。

アプリケーションをプロファイルする場合、通常はカーネルをプロファイルする必要はありませんので、`--no-vmlinux` オプションを指定して情報量を減らしておくことをお勧めします。

8.4.1 一般的な手順

もっともシンプルな方法でプロファイルを行なう場合は、まずデーモンを起動してデータを収集し、デーモンを停止してレポートを作成する、という流れになります。この方法をより詳しく書くと、下記のようになります:

- 1 シェルを起動して `root` でログインします。
- 2 Linux カーネルを利用してプロファイルするか、もしくは利用せずにプロファイルするかを決めます:

2a Linux カーネルを利用するプロファイル 下記のコマンドを実行します。これは `opcontrol` コマンドが非圧縮のイメージファイルを利用するためです:

```
cp /boot/vmlinux-`uname -r`.gz /tmp
gunzip /tmp/vmlinux*.gz
opcontrol --vmlinux=/tmp/vmlinux*
```

2b Linux カーネルを利用しないプロファイル 下記のコマンドを実行します:

```
opcontrol --no-vmlinux
```

ある関数から別の関数を呼び出す様子を出力に含めたい場合は、`--callgraph` オプションと最大の呼び出し履歴数 (*DEPTH*) を追加で指定します:

```
opcontrol --no-vmlinux --callgraph DEPTH
```

3 OProfile デーモンを起動します:

```
opcontrol --start
Using 2.6+ OProfile kernel interface.
Using log file /var/lib/oprofile/samples/oprofiled.log
Daemon started.
Profiler running.
```

4 前項の作業が完了したら、プロファイル対象のアプリケーションを起動します。

5 OProfile デーモンを停止します:

```
opcontrol --stop
```

6 収集したデータを /var/lib/oprofile/samples 以下に出力します:

```
opcontrol --dump
```

7 レポートを作成します:

```
opreport
Overflow stats not available
CPU: CPU with timer interrupt, speed 0 MHz (estimated)
Profiling through timer interrupt
      TIMER:0|
samples|      %|
-----|-----
      84877 98.3226 no-vmlinux
...
```

8 OProfile デーモンをシャットダウンします:

```
opcontrol --shutdown
```

8.4.2 イベント設定の取得

イベントの設定を行なうための一般的な手順は下記の通りです:

- 1 最適化のヒントを得るには、まず CPU_CLK_UNHALTED と INST_RETIRED のイベントを指定して使用します。
- 2 ボトルネックを検出するため、特定のイベントを指定して使用します。イベントの一覧を表示するには、`opcontrol --list-events` を実行してください。

特定のイベントに対してプロファイルを行ないたい場合は、まず `ophelp` コマンドを利用して、お使いのプロセッサが そのイベントに対応しているかどうかを確認してください (下記は Intel Core i5 CPU での出力例です):

```
ophelp
```

```
oprofile: available events for CPU type "Intel Architectural Perfmon"
```

```
See Intel 64 and IA-32 Architectures Software Developer's Manual
Volume 3B (Document 253669) Chapter 18 for architectural perfmon events
This is a limited set of fallback events because oprofile doesn't know your CPU
CPU_CLK_UNHALTED: (counter: all))
    Clock cycles when not halted (min count: 6000)
INST_RETIRED: (counter: all))
    number of instructions retired (min count: 6000)
LLC_MISSES: (counter: all))
    Last level cache demand requests from this core that missed the LLC (min count:
6000)
    Unit masks (default 0x41)
    -----
    0x41: No unit mask
LLC_REFS: (counter: all))
    Last level cache demand requests from this core (min count: 6000)
    Unit masks (default 0x4f)
    -----
    0x4f: No unit mask
BR_MISS_PRED_RETIRED: (counter: all))
    number of mispredicted branches retired (precise) (min count: 500)
```

なお、`opcontrol--list-events` コマンド でも同じ出力を得ることができます。

上記で対応していることを確認したら、`--event` オプションで イベントを指定して実行します。なお、複数のオプションを指定することもできます。また、このオプションではイベント名 (ophelp での出力) と 採取間隔 (サンプルレート) を指定します。たとえば下記のようになります:

```
opcontrol --event=CPU_CLK_UNHALTED:100000
```

警告: CPU_CLK_UNHALTED の採取間隔によってもたらされる負荷について

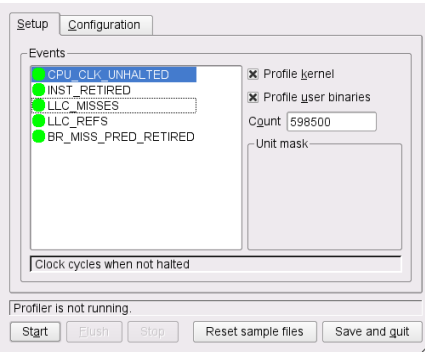
短い時間間隔でサンプリングを行なうと、システムに高い負荷をかけることになってしまうほか、場合によってはフリーズしてしまうことがあります。

8.5 OProfile GUI の使用

OProfile 向けの GUI は、`root` から `oprof_start` コマンドで起動することができます (図8.1「OProfile 向け GUI」(106 ページ) のように表示されます)。あとは採取したいイベントを選択し、必要であれば カウンタを変更してください。するとチェック対象のイベントの一覧に、緑色の行が追加されます。また、それらの行にマウスカーソルを合わせると、ステータス行にヘルプメッセージが表示されます。*Configuration* (設定) タブを利用すると、バッファと CPU サイズ、冗長出力オブ

ションなどを設定することができます。最後に *Start* (開始) を押すと、OProfile を実行することができます。

図 8.1 OProfile 向け GUI



8.6 レポートの作成

レポートを生成する前に、まずは OProfile が `/var/lib/oprofile/samples` ディレクトリ内にダンプを出力していることを確認してください。なお、ダンプ出力は `opcontrol --dump` コマンドで行なうことができます。レポートは、それぞれ `opreport` または `opannotate` のコマンドで行なうことができます。

`opreport` をパラメータ無しで実行すると、完全な概要情報を出力します。パラメータとして実行ファイルを指定すると、その実行ファイルからの情報のみを出力します。C++ 言語で書かれたアプリケーションを分析する場合は、`--demangle smart` オプションをご利用ください。

一方の `opannotate` コマンドでは、ソースコードから注釈情報を出力します。下記のようなオプションを指定して実行します：

```
opannotate --source ¥
--base-dirs=BASEDIR ¥
--search-dirs= ¥
--output-dir=annotated/ ¥
/lib/libfoo.so
```

`--base-dir` オプションには、実行ファイル内のデバッグ情報が利用するソースコードのパスを、カンマ区切りで指定します。ここで指定するパスは `--search-dirs` での指定よりも前に使用します。 `--search-dirs` オプションは、ソースコードを検索するディレクトリを、カンマ区切りで指定します。

注記: 注釈付きソースの不確実性について

コンパイラの最適化機能を設定すると、そのときのオプション設定やソースコードの状態によっては、部分的にコードが削減されたり追加されたりする場合があります。<http://oprofile.sourceforge.net/doc/debug-info.html> にある情報をお読みになり、どのようなことが起こるのかを知っておいてください。

8.7 さらになる情報

この章では大まかな説明のみを行なってきました。より詳しい情報については、それぞれ 下記のリンク先 (それぞれ英語です) をお読みください:

<http://oprofile.sourceforge.net>

プロジェクトの Web ページです。

マニュアルページ

各種ツールのオプションについて、詳しい説明が書かれています。

</usr/share/doc/packages/oprofile/oprofile.html>

OProfile のマニュアルが用意されています。

<http://developer.intel.com/>

Intel プロセッサのアーキテクチャ仕様が書かれています。

[http://www.amd.com/us-en/assets/content_type/](http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/22007.pdf)

[white_papers_and_tech_docs/22007.pdf](http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/22007.pdf)

AMD Athlon/Opteron/Phenom/Turion 向けのアーキテクチャ仕様が書かれています。

<http://www-01.ibm.com/chips/techlib/techlib.nsf/productfamilies/>

PowerPC/

IBM iSeries, pSeries, ブレードサーバシステムにおける、PowerPC64 プロセッサのアーキテクチャ仕様が書かれています。

パート IV. リソース管理

一般的なシステムリソース管理

システムのチューニング (最適化) では、カーネルの最適化やお使いのアプリケーションの 設定だけでなく、無駄のない高速なシステムを構築することをも意味します。たとえばパーティション設定やファイルシステムの設定次第では、サーバの動作速度が より高速化されたりする場合があります。また、有効に設定したサービスの数や定常的な 作業方法なども性能に影響することがあります。

9.1 インストールの計画

インストール作業時に注意深く設定を行なうことで、特定の目的で使用する際に必要な 要件のうち、基本的な部分を正確に構築することができます。これはシステムのチューニングにかかる手間を省くことにも繋がります。この章で提案する様々な設定変更 は、それぞれインストール時の *インストール設定* で行ないます。詳しくは 項「インストール設定」(第1章 *YaST を利用したインストール*, ↑リファレンス) をお読みください。

9.1.1 パーティション設定

サーバで使用するアプリケーションの範囲と、ハードウェア構成に依存しますが、パーティション方法はマシンの性能に影響します (ただし悪い方向のみですが)。特定の用途に対しては異なるパーティション方法を行なうべきである、ということはこのマニュアルでは詳しく説明していませんが、それぞれ下記の要件を満たすことでよりよい方向に性能を改善することができます。もちろん外付けのストレージシステム の場合には、下記の要件を満たす必要はありません。

- ディスク上には、常にある程度の空き容量を残してください。ディスクを満杯にしていると、性能を落とすことに繋がってしまいます。
- たとえば下記のような方法で、読み書きのアクセスを異なるディスクに分散させてください：
 - オペレーティングシステム自身とデータ、ログファイルなどについて、それぞれ別々のディスクを使用するように設計する
 - メールサーバのスパール (spool) ディレクトリを、個別のディスクに配置する
 - ユーザのホームディレクトリを、複数のディレクトリに分散させる

9.1.2 インストールの範囲

実際にはインストールの範囲そのものが直接的にマシンの性能に影響することはありませんが、注意深くパッケージ範囲を選択することには十分な意味があります。サーバを動作させるにあたって、最小限のパッケージのみをインストールしてください。パッケージの選択を最小限に絞ることは、管理の手間を減らす意味があるほか、潜在的なセキュリティ問題を回避することにも繋がります。また、それぞれのサーバに対して別々の構築を行なうことで、既定では不要なサービスを開始しないことにもなります。

openSUSE ではインストールの概要画面でインストール範囲を設定することができます。既定では特定の用途向けに設定されたパターンを選択するだけですが、YaST のソフトウェア管理モジュールを利用して、パッケージ単位での選択を行なうこともできます。

サーバとして使用するにあたっては、それぞれ下記のパターンが不要と思われます：

GNOME デスクトップ環境

サーバとして動作する場合は、いわゆる GUI を必要とすることはほとんどありません。GUI が必要であったとしても、icewm や fwm などより経済的な選択を行なったほうが適切です。

X ウィンドウシステム

サーバの管理とコマンドラインでのアプリケーション実行を行なうだけであれば、このパターンをインストールしないことをお勧めします。ただし、GUI アプリケーションをネットワーク上で離れた (リモートの) マシンから実行する場合であっても、GUI のインストールは必要となることに注意してください。たとえば

お使いのアプリケーションが GUI からの管理しか行なえないようなものであったり、YaST の GUI 版を使用したりしたい場合は、このパターンをインストールしておいてください。

印刷サーバ

このパターンは、そのマシンから印刷を行なう場合にのみ必要となるものです。

9.1.3 既定のランレベル

X Window System を実行すると多くの資源を占有してしまうものですが、サーバを稼働させるにあたって必ずしも必要なものではありません。そのため、お使いのシステムがサーバ用途である場合、ランレベルを 3 (X Window System を動作させない、ネットワーク付きマルチユーザモード) に設定することをお勧めします。もちろんこの場合でもリモートから GUI アプリケーションを起動することはできますし、startx コマンドで GUI デスクトップをローカルのコМПユータ上で起動することもできます。

9.2 不要なサービスの無効化

既定のインストールでは、いくつかのサービス (どのサービスが起動されるかは、インストール範囲に依存します) が起動されます。それぞれのサービスはある程度の資源を占有するものであるため、不要なサービスについては停止させておくことをお勧めします。YaST > システム > システムサービス (ランレベル) > 熟練者モードを選択し、サービスを管理するモードに移動してください。YaST のグラフィカル (GUI) 版を利用している場合は、列ヘッダをマウスの左ボタンで押すことで、並べ替えを行なうことができます。この一覧ではサービスが現時点で動作中であるかどうかと、サーバの既定のランレベルでどのサービスが起動されるのかが表示されます。各サービスの説明を読みたい場合は、それぞれのサービスをマウスの左ボタンで選択してください。また、開始／停止／更新のドロップダウンボタンを押すと、それぞれサービスをすぐに開始したり停止したりすることができます。恒久的に無効化したい場合は、セット／リセットのドロップダウンをお使いください。

下記には、openSUSE を既定の状態でインストールした場合に有効化されるサービスのうち、不要と思われるものを列挙します:

alsasound

Advanced Linux Sound System を読み込みます。音声の再生機能を必要としない場合は無効化することができます。

auditd

監査システム向けのデーモンです。監査を必要としない場合は無効化することができます。

bluez-coldplug

Bluetooth デバイスのコールドプラグ (コンピュータの電源が落とされている時に 抜き差しすること) を処理するためのものです。Bluetooth をお持ちでない場合は無効化することができます。

cups

印刷デーモンです。プリンタをお持ちでない場合は無効化することができます。

java.binfmt_misc

*.class や *.jar の形式で提供される Java プログラムを実行するためのものです。Java アプリケーションを実行しない場合は、無効化することができます。

nfs

NFS ファイルシステムをマウントする際に必要なサービスです。不要であれば無効化することができます。

smbfs

Windows サーバが提供する SMB/CIFS ファイルシステムをマウントするのに必要なサービスです。不要であれば無効化することができます。

splash / splash_early

起動時のスプラッシュスクリーンを表示するためのものです。サーバとして動作させる 場合、通常は不要なものです。

9.3 ファイルシステムとディスクアクセス

ハードディスクはコンピュータシステム内で最も遅いデバイスであるため、よくボトルネックの原因となります。性能を改善するため、お使いの作業内容に応じて適切なファイルシステムをご使用ください。また、特殊なマウントオプションを使用したり、プロセスの I/O 優先順序を設定したりすることで、さらなる高速化を行なうこともできます。

9.3.1 ファイルシステム

openSUSE は複数のファイルシステムを提供しています。それぞれ btrfs, ext3, ext2, reiserfs, xfs などと呼ばれています。それぞれのファイルシステムにはそれぞれの 利点と欠点が存在しています。

9.3.1.1 NFS

NFS (バージョン 3) のチューニングについては、NFS の Howto ドキュメント (<http://nfs.sourceforge.net/nfs-howto/>) 内に詳しい説明が あります。なお、NFS 共有のマウントを行なう際は、マウントオプション `wsiz` と `rsiz` を使用して、それぞれ 読み書きのブロックサイズを 32768 に設定してみることをお勧めします。

9.3.2 アクセス日時 (atime) 更新の無効化

ファイルを Linux のファイルシステムから読み込む場合、対象のファイルのアクセス日時 (atime) が更新されます。その結果、読み込みのみのアクセスを行なった場合でも、書き込みの処理が発生することになります。さらに言うと、ジャーナル機能 付きのファイルシステムの場合は、ジャーナル側も更新する必要があるため、2 回の書き込み処理が発生します。この機能を無効化し、アクセス日時を更新しないようにすることで、性能を改善することができます。これは特にファイルサーバや Web サーバ、ネットワークストレージなどの形態で使用している場合、性能改善に繋がります。

アクセス日時の更新を無効化するには、ファイルシステムのマウント時に `noatime` オプションを設定します。これを行なうには `/etc/fstab` を直接編集するか、もしくは YaST パーティション設定で、編集または追加を行なう際に `Fstab` オプション で設定を行ないます。

9.3.3 ionice を利用したディスクアクセスの優先順位設定

`ionice` コマンドは、特定のプロセスに対してディスクのアクセス 優先順位を設定するためのものです。このコマンドを利用することで、たとえばバックアップ ジョブなど、ディスク負荷が高く時間に余裕のある処理に対して、低い優先順位を設定したり することができます。また、`ionice` コマンドは特定のプロセスに 対して優先順位を高く設定し、ディスクへのアクセスを即時に行なわなければならない処理に 対応する

こともできます。それぞれ下記の 3 種類のスケジュール区分を設定することができます:

アイドル

スケジュール区分がアイドルに設定されたプロセスは、他のプロセスがディスク入出力を必要としていない場合にのみ、ディスクへのアクセスを行なうことができます。

ベストエフォート

既定では、全てのプロセスがこの I/O 優先順位に設定されます。この区分では、さらにレベルが 0 から 7 までに細分化されていて、0 が最も高い優先順位になっています。既定では CPU の nice 値に合わせてプロセスの I/O 優先順位が設定されます。

リアルタイム

この区分に属するプロセスは、常にディスクへのアクセスを優先的に獲得することができます。この区分でも、さらに 0 から 7 までのレベルに細分化されていて、0 が最も高い優先順位になります。なお、この区分を利用すると、他のプロセスが全くディスクにアクセスできなくなってしまう場合があります。ことに注意してください。

ionice について、詳しいコマンド書式や説明を読むには、ionice(1) のマニュアルページをお読みください。

カーネルのコントロールグループ

10

カーネルのコントロールグループ (「cgroups」と略して表記します) は、タスク (プロセス) をまとめたり分割したりすることのできるカーネルの機能で、子プロセスを階層構造でまとめたグループとして扱うこともできる仕組みです。これらの階層構造は、利用可能なハードウェアやネットワーク資源を有効に活用するため、システムチューニングを支援するよう設定することができます。

10.1 技術概要と用語定義

本章では下記のような用語を使用します:

- 「cgroup」はコントロールグループの略称です。
- cgroup 内では複数のタスク (プロセス) が関連づけられていて、それらのタスク向けに構成されたパラメータをもとに動作する、各種のサブシステムが存在しています。
- サブシステムは cpuset や freezer、メモリやディスクの I/Oなどを割り当てたり設定したりするパラメータを提供します。より一般的な書き方をすると、「資源コントローラ」と呼びます。
- cgroup は木構造で階層管理されています。システム内には複数の木構造を持つことができるようになっていて、特定の環境に適応するよう異なる木構造を利用することもできます。
- システム内で動作するすべてのタスクは、必ずいずれかの cgroup の木構造に属します。

10.2 シナリオ

理解を深めるため、下記の資源計画シナリオをお読みください (ソース: /usr/src/linux/Documentation/cgroups/cgroups.txt):

図 10.1 資源計画

Firefox のような Web ブラウザの場合は Web ネットワーククラスに分類されますし、(k)nfsd のような NFS デーモンであれば NFS ネットワーククラスに分類されます。一方、Firefox は誰がそれを起動したのかに依存して、適切な CPU またはメモリクラスを共有します。

10.3 コントロールグループサブシステム

それぞれ下記のようなサブシステムが存在していて、それらは 2 種類に分類することができます:

分離と特殊コントローラ

cpuset, freezer, devices, checkpoint/restart

資源コントローラ

cpu (スケジューラ), cpuacct, メモリ, ディスク I/O, ネットワーク

サブシステムのマウントは、下記のようにして別々に行なうこともできます:

```
mount -t cgroup -o cpu none /cpu
mount -t cgroup -o cpuset none /cpuset
```

一括で行なう場合は下記のようにします。なお、デバイス名に (下記で none と設定されている箇所) ついては、任意の名前を設定することができます。ここで指定した名前は、/proc/mounts で表示することができます:

```
mount -t cgroup none /sys/fs/cgroup
```

サブシステムに関する追加情報は下記の通りです:

cpuset (分離)

cpuset を使用することで、プロセスをシステム内の CPU のサブセットやメモリ (「メモリノード」) に結びつけることができます。たとえば 10.4.3 項「例: cpuset」(122 ページ) をお読みください。

freezer (制御)

freezer サブシステムは、高性能計算機クラスタ (HPC クラスタ) を利用する際に 便利な仕組みで、事前に設定されたチェックポイントに到達した際にグループ内にあるすべてのタスクをフリーズ (停止) させたりタスク群を停止させたりすることができるものです。詳しくは `/usr/src/linux/Documentation/cgroups/freezer-subsystem.txt` をお読みください。

下記に示すのは、freezer サブシステムの使用方法を説明するための 基本的なコマンド群です:

```
mount -t cgroup -o freezer freezer /freezer
# 子 cgroup を作成:
mkdir /freezer/0
# 作成した cgroup にタスクを投入する:
echo $task_pid > /freezer/0/tasks
# フリーズさせる:
echo FROZEN > /freezer/0/freezer.state
# フリーズ状態を解除する:
echo THAWED > /freezer/0/freezer.state
```

Checkpoint/Restart (制御)

cgroup に属するすべてのプロセスに対して、それらの状態をダンプファイルに保存します。これを利用することで、後から実行を再開することができます (もちろん保存後にそのまま実行し続けることもできます)。

また、「保存されたコンテナ」について、物理マシン間を 起動することもできます (仮想マシンで行なうことができるのと同じ機能を 提供します)。

すべてのプロセスのイメージは、ファイルに出力されます。

デバイス (分離)

システム管理者は、cgroup 内のプロセスからアクセスできるデバイスについて、その一覧を設定することができます。

これにより、そのデバイスに対してのアクセスを指定した cgroup に属するタスクに対してのみ許可するように制限することができます。詳しくは `/usr/src/linux/Documentation/cgroups/devices.txt` をお読みください。

cpuacct (制御)

cpuacct は CPU の使用量を計算するコントローラで、cgroup を利用することでタスクをグループ化し、それらのグループに対する CPU 使用率を計算します。詳しくは `/usr/src/linux/Documentation/cgroups/cpuacct.txt` をお読みください。

CPU (資源制御)

CFS (スケジューラ) のグループスケジュール機能を利用して、グループ間での CPU 帯域の共有を行ないます。かなり複雑な処理を行ないます。

メモリ (資源制御)

- ユーザスペースのプログラムに対して、メモリの使用量を制限します。
- カーネルの起動パラメータに対して `swapaccount=1` を指定すると、スワップの使用方法を制御することができます。
- LRU (Least Recently Used; 使用率の低い) ページを制限します。
- 匿名ページとファイルキャッシュを対象とします。
- カーネルのメモリに対しては制限を行ないません。
- 必要であれば他のシステムでも利用されます。

詳しくは `/usr/src/linux/Documentation/cgroups/memory.txt` をお読みください。

ブロック I/O (資源制御)

blkio (ブロック IO) コントローラは、ディスクの I/O コントローラとして利用できる仕組みです。blkio コントローラでは、ディスクの使用帯域を制御するためのポリシーを設定することができます。

blkio.weight の値で重さ (weight) を設定し、帯域の重み付けを設定する場合は、基本コマンドで設定することができます：

```
# /sys/fs/cgroup 内の準備
mkdir /sys/fs/cgroup/blkio
mount -t cgroup -o blkio none /sys/fs/cgroup/blkio
# 2 つの cgroup の開始
mkdir -p /sys/fs/cgroup/blkio/group1 /sys/fs/cgroup/blkio/group2
# 重み付けの設定
echo 1000 > /sys/fs/cgroup/blkio/group1/blkio.weight
echo 500 > /sys/fs/cgroup/blkio/group2/blkio.weight
# それぞれのグループに対して、制御下に置くプロセスの
# PID を書き込む
command1 &
echo $! > /sys/fs/cgroup/blkio/group1/tasks

command2 &
echo $! > /sys/fs/cgroup/blkio/group2/tasks
```

下記の例は、`blkio.throttle.read_bps_device` で読み込み量の上限を、`blkio.throttle.write_bps_device` で書き込み量の制限を行なう場合の基本的なコマンド例です:

```
# /sys/fs/cgroup 内の準備
mkdir /sys/fs/cgroup/blkio
mount -t cgroup -o blkio none /sys/fs/cgroup/blkio
# ルートグループに対するデバイスの帯域幅の指定; 書式:
# <major>:<minor> <bytes_per_second>
echo "8:16 1048576" > /sys/fs/cgroup/blkio/blkio.throttle.read_bps_device
```

注意事項やシナリオ、追加のパラメータについて、詳しくは `/usr/src/linux/Documentation/cgroups/blkio-controller.txt` をお読みください。

ネットワーク I/O (資源制御) (Draft)
現在議論中。

10.4 コントローラグループの使用

10.4.1 要件

`cgroup` を便利に使用するには、下記の追加パッケージをインストール してください:

- `libcgroup1`: 資源管理を簡略化するための 基本的なユーザスペースツールが含まれています。
- `cpuset`: CPU セットを管理するための `cset` ツールが含まれています。
- `libcpuset1`: CPU セットを操作するための C 言語 API が含まれています。
- `kernel-source`: ドキュメントを読む場合に 必要なパッケージです。
- `lxc`: Linux のコンテナ実装です。

10.4.2 環境の確認

`openSUSE` に同梱されているカーネルであれば、`cgroup` に対応しています。特に追加の修正を適用する必要はありません。`lxc-checkconfig` を実行すると、下記のような形式で `cgroup` の環境を表示することができます:

```
--- Namespaces ---
Namespaces: enabled
Utsname namespace: enabled
Ipc namespace: enabled
Pid namespace: enabled
User namespace: enabled
Network namespace: enabled
Multiple /dev/pts instances: enabled
```

```
--- Control groups ---
Cgroup: enabled
Cgroup namespace: enabled
Cgroup device: enabled
Cgroup sched: enabled
Cgroup cpu account: enabled
Cgroup memory controller: enabled
Cgroup cpuset: enabled
```

```
--- Misc ---
Veth pair device: enabled
Macvlan: enabled
Vlan: enabled
File capabilities: enabled
```

どのサブシステムが利用可能なのかを知るには、下記を実行します:

```
mkdir /cgroups
mount -t cgroup none /cgroups
grep cgroup /proc/mounts
```

それぞれ下記のサブシステムが利用できます: perf_event, blkio, net_cls, freezer, devices, memory, cpuacct, cpu, cpuset.

10.4.3 例: cpuset

コマンドラインを利用する場合は下記のようにします:

- 1 CPU 数を判別したり、メモリノードを確認したりするには、それぞれ /proc/cpuinfo や /proc/zoneinfo ファイルの内容を表示します。
- 2 仮想ファイルシステム内に cpuset 用の構造を作成します (情報源: /usr/src/linux/Documentation/cgroups/cpusets.txt):

```
mount -t cgroup -ocpuset cpuset /sys/fs/cgroup/cpuset
cd /sys/fs/cgroup/cpuset
mkdir Charlie
cd Charlie
# この cpuset で使用する CPU の一覧を設定します:
echo 2-3 > cpuset.cpus
# この cpuset で使用するメモリノードを設定します:
```



```
echo 1 > cpuset.mems
echo $$ > tasks
# これでサブシェル 'sh' が Charlie という名前の cpuset に
# 設定されます。下記を実行すると '/Charlie' と表示されるはずです:
cat /proc/self/cpuset
```

3 cpuset の削除は下記のようなシェルコマンドを実行することで行ないます:

```
rmdir /sys/fs/cgroup/cpuset/Charlie
```

上記のコマンドは、cpuset が使用中の場合には失敗します。削除を行なう前に、それらに属する内部 cpuset やタスク (プロセス) を取り除いてください。下記のように実行すると、それらを確認することができます:

```
cat /sys/fs/cgroup/cpuset/Charlie/tasks
```

背景となる情報や追加の設定フラグについて、詳しくは `/usr/src/linux/Documentation/cgroups/cpusets.txt` をお読みください。

cset ツールを利用する場合は下記のようにします:

```
# CPU 数とメモリノードの表示
cset set --list
# cpuset 構造の作成
cset set --cpu=2-3 --mem=1 --set=Charlie
# cpuset 内でプロセスを起動
cset proc --set Charlie --exec -- stress -c 1 &
# 既存のプロセスを cpuset に移動
cset proc --move --pid PID --toset=Charlie
# cpuset 内のタスク一覧を表示
cset proc --list --set Charlie
# cpuset の削除
cset set --destroy Charlie
```

10.4.4 例: cgroup

コマンドラインを利用する場合は下記のようにします:

1 cgroup の構造を作成します:

```
mount -t cgroup cgroup /sys/fs/cgroup
cd /sys/fs/cgroup/cpuset/cgroup
mkdir priority
cd priority
cat cpu.shares
```

2 cpu.shares の値は下記のような意味があります:

- 1024 が既定値 (詳しくは /Documentation/scheduler/sched-design-CFS.txt を参照) = 50% の使用率
- 1524 = 60% の使用率
- 2048 = 67% の使用率
- 512 = 40% の使用率

3 cpu.shares を変更します:

```
echo 1024 > cpu.shares
```

10.5 さらなる情報

- カーネルのドキュメンテーション: (kernel-source パッケージ内) /usr/src/linux/Documentation/cgroups ディレクトリ内:
 - /usr/src/linux/Documentation/cgroups/blkio-controller.txt
 - /usr/src/linux/Documentation/cgroups/cgroups.txt
 - /usr/src/linux/Documentation/cgroups/cpuacct.txt
 - /usr/src/linux/Documentation/cgroups/cpusets.txt
 - /usr/src/linux/Documentation/cgroups/devices.txt
 - /usr/src/linux/Documentation/cgroups/freezer-subsystem.txt
 - /usr/src/linux/Documentation/cgroups/memcg_test.txt
 - /usr/src/linux/Documentation/cgroups/memory.txt
 - /usr/src/linux/Documentation/cgroups/resource_counter.txt
- <http://lwn.net/Articles/243795/>—Corbet, Jonathan: Controlling memory use in containers (2007).
- <http://lwn.net/Articles/236038/>—Corbet, Jonathan: Process containers (2007).

電源管理

電源管理とは、現状のシステムにおける要件を満たしたまま、エネルギー消費にかかるコストを減らし、同時にシステムの冷却を行なうための仕組みです。そのため、電源管理はシステムに対する実際の性能要件と省電力のバランスを追い求める問題になります。電源管理はシステムの様々な箇所で実施することができます。デバイスに対する電源管理機能の仕様や、その電源管理機能に対するオペレーティングシステム向けのインターフェイスの集合体は、Advanced Configuration and Power Interface (ACPI) として規定されています。サーバ用途における省電力は主にプロセッサ上で実現できるものであるため、本章では省電力に関する複数の分析ツールや、関連パラメータの影響について紹介しています。

11.1 CPU における電源管理

CPU では様々な方法で電源消費を制御することができます。たとえばプロセッサを一時的に待機させたり (C-ステート)、CPU の動作周波数を変更したり (P-ステート) することができるほか、CPU の減速を設定したり (T-ステート) することもできます。下記の章では、それぞれのアプローチにおける概要説明と、それらの重要性を説明します。より詳しい仕様説明をご希望の場合は、<http://www.acpi.info/spec.htm> をお読みください。

11.1.1 C-ステート (プロセッサの動作設定)

新しいプロセッサであれば、C-ステートと呼ばれる省電力モードに対応しています。これは、使用されていない CPU 内コンポーネントの電源を落とすことで省電力を実現するものです。C-ステートは、その昔、ラップトップ機のみで利用できる機能で

したが、サーバ上でも使用できるようになっています (たとえば Intel* プロセッサでは、Nehalem と呼ばれる世代のプロセッサから利用できるようになっています)。

プロセッサが C0 ステートで動作している場合、これは 何らかの命令を実行していることを示しています。その他の C-ステートの場合は 特に実行すべき命令が与えられていないことを示しています。また、C に続く番号 がより高いほど、より深い待機状態にあることを示し、省電力のためにより多くの コンポーネントが待機状態に置かれます。深い待機状態ほどより省電力になります が、待機状態が深ければ深いほど復元時の遅延が増大 (つまり C0 ステートに戻るまでの時間が長く) になります。

ステートによっては、さらに細かい省電力レベルを提供する目的で、サブモードを 提供している場合があります。どの C-ステートやサブモードに対応しているかは プロセッサによって異なりますが、C1 については常に 対応しています。

表11.1「C-ステート」(126 ページ) では、一般的に使用されている C-ステートについて説明を示しています。

表 11.1 C-ステート

モード	説明
C0	動作モードです。CPU のすべてのコンポーネントに電源が投入されている状態です。
C1	第一段階の待機状態です。ソフトウェアを介した CPU の内部クロックが停止します。バスインターフェイスと APIC はフルスピードで動作し続けます。
C2	ハードウェアを介した CPU の内部メインクロックが停止します。プロセッサが管理する、ソフトウェアから閲覧可能な状態は保持されますが、割り込みによる待機状態の解除には、より長い時間がかかります。
C3	すべての CPU 内部クロックを停止します。プロセッサは自身のキャッシュについての一貫性を保持しなくな

モード	説明
	りますが、その他の状態は管理し続けます。プロセッサによっては C3 ステートに複数のバリエーションが用意されていて、割り込みによる待機状態解除にかかる時間がそれぞれ異なります。

11.1.2 P-ステート (プロセッサの性能設定)

プロセッサが動作状態にある (C0 ステート) 場合、CPU に対して性能設定 (P-ステート) を行なうことができます。C-ステートが 待機状態を設定するのに対して、P-ステート は CPU の 動作周波数と電圧に関する性能設定を行ないます。

P に続く数値が大きければ大きいほど、より低い周波数と低い電圧でプロセッサが動作することになります。また、P に続く数値はプロセッサ固有のもので、どれだけの周波数と電圧になるのかは、プロセッサによって様々です。ただし P0 は常に最高の性能を提供する状態を指します。また、P-ステートの数値が大きければ大きいほどプロセッサの速度が低くなり、それとともに電力消費も小さくなります。たとえば P3-ステートにあるプロセッサは、P1-ステートにあるプロセッサに比べて動作が遅く、電力消費も小さな状態です。また、何らかの P-ステートで動作するには、プロセッサが動作中である C0-ステートでなければなりません。なお、CPU の P-ステートは Advanced Configuration and Power Interface (ACPI) 仕様 (<http://www.acpi.info/spec.htm>) で規定されています。

C-ステートと P-ステートはそれぞれ独立した仕組みです。

11.1.3 T-ステート (プロセッサの減速設定)

T-ステートとは、プロセッサの温度上昇を防ぐためにその動作周波数を遅くする 状態を指す言葉です。これは固定の割合で CPU の動作を強制的に減速させることで実現します。減速設定は T1 (CPU に対して強制的な 減速を行なわない状態) から T n までの範囲が存在していて、それぞれ n の数値が 大きければ大きいほど、減速する割合が大きくなります。

なお、減速設定では電圧の調整を行なうことはなく、CPU を一定時間だけ待機させる 仕組みであるため、処理を完了するまでの時間が長くなるほか、省電力という意味では逆効果になることに注意してください。

また、T-ステートは主に温度上昇を防ぐ効果を期待して利用するものです。T-ステートは C-ステートの移行 (より高い C-ステートへの移行) を阻害する場合があります。そのため、C-ステートに対応した CPU の場合は、かえって消費電力を増やしてしまう場合があります。

11.1.4 ターボ機能

ある時期から、CPU での電源消費と性能のチューニングは、もはや周波数制御 だけの話ではなくなっていました。新しいプロセッサの場合、これに加えて 性能と省電力の両方についてバランスを保つために、深いスリープ状態と従来の 周波数制御が使用され、場合によっては周波数を上げる場合もあるよう になりました。最新の AMD* および Intel* プロセッサでは、ターボ機能 (Turbo CORE* や Turbo Boost* と呼ばれています) が提供され、特定の CPU コアの周波数を動的に増加させる一方、その他のコアをより深いスリープ状態に 移行させるようになっています。これにより、有効なスレッドに対する性能を 上げることができる一方、熱設計電力 (Thermal Design Power (TDP)) を維持 することができるようになっています。

しかしながら、どの CPU コアがターボ機能を利用して周波数を上げるのかに ついては、アーキテクチャに大きく依存します。このような機能を有効に活用する ための 方法については、11.3.1 項「cpupower ツールの使用」(132 ページ) をお読みく ださい。

11.2 Linux カーネルの CPUfreq インフラストラクチャ

プロセッサの性能設定 (P-ステート) とプロセッサの動作設定 (C-ステート) は、それぞれプロセッサを異なる動作周波数に切り替えたり、電源消費を抑えるために 電圧を調整したりする機能です。

システムの稼働中、動的にプロセッサの周波数を変更するにあたっては、CPUfreq インフラストラクチャを利用して、静的または動的な電源ポリシーを設定することが できます。ここで使用される主なコンポーネントは CPUfreq サブシステム (ローレベルの技術仕様から、ハイレベルのポリシー設定に至るまでの汎用的な インターフェイスを提供するサブシステム) のほか、カーネル内蔵の調整器 (様々な 判断条件で CPU の周波数を変更するポリシーベースの調整器) や、それらの技術に 対応するための CPU 固有のドライバなどがあります。

動作速度を動的に調整して速度を落とすことで省電力を実現することができるほか、発熱量を減らすこともできます。

11.2.1 カーネル内蔵の調整器

あらかじめ用意された CPU 電源制御の方式の 1 つとして、カーネル内蔵の調整器 (ガバナーとも呼びます) を利用することができます。CPUfreq の調整器は、周波数の変更と省電力の調整を行なう際に P-ステートを利用します。動的な調整器は、CPU に対して必要とされる性能を満たす範囲で、動的に CPU の周波数を切り替えます。これらの調整器はカスタマイズを行なって細かい調整を行なうことができるようになっていきます。

CPUfreq サブシステムでは、下記の調整器を利用することができます：

性能重視型調整器 (Performance Governor)

最大限の性能を発揮するため、CPU の動作周波数は最高値に固定されます。そのため、この調整器では省電力への考慮は行なわれないということになります。

チューニングオプション: 調整器が使用する最大の周波数範囲を設定することができます (たとえば `cpupower` コマンドラインツールを利用して行なうことができます)。

省電力重視型調整器 (Powersave Governor)

CPU の動作周波数は最低値に固定されます。この調整器は、プロセッサの負荷に関係なく周波数が固定されて変更されないことから、性能面への影響が最も大きく存在する調整器になります。

ただし、この調整器を利用しても期待通りの省電力を実現できない場合もあります。これは一般的に、C-ステートを利用した一時休止のほうがよりよい省電力を実現できるためです。また、省電力重視型調整器を利用して動作周波数を最小限の値に固定すると、それだけ必要な処理を行なうのに時間がかかることになるため、システムが C-ステートに入るまでの時間も長くなります。

チューニングオプション: 調整器が使用する最小の周波数範囲を設定することができます (たとえば `cpupower` コマンドラインツールを利用して行なうことができます)。

オンデマンド型調整器 (On-demand Governor)

動的に CPU 動作周波数を調整するカーネル実装ポリシーです。調整器はプロセッサの使用率を監視し、特定の閾 (しきい) 値を超えると調整器が動作し、

利用可能な最大の周波数に設定します。使用率が特定の閾値を下回ると、周波数を 1 段階下げます。さらに使用率が低いままである場合は、最小値になるまで段階的に周波数を下げ続けます。

openSUSE では、オンデマンド型の調整器が既定値になっていて、もっともよくテストされています。

チューニングオプション: 調整器が使用する周波数範囲と、調整器が使用率をチェックする頻度、および調整を行なう際の使用率の閾値をそれぞれ設定することができます。またオンデマンド型調整器では、`ignore_nice_load` と呼ばれる値を調整することもできます。詳しくは 手順 11.1「プロセッサ使用率における nice 値の無視」(137 ページ) をお読みください。

保守型調整器 (Conservative Governor)

この調整器はオンデマンド型のものに似た仕組みで、プロセッサの使用率に応じて動作周波数を調整します。ただし周波数を上げる場合には、より段階的な変更を行なうようになっています。プロセッサの使用率が特定の閾値を超えると、(オンデマンド型の調整器が行なうように) すぐに最大値を設定することはなく、次に設定可能な高い周波数を設定するようになっています。

チューニングオプション: 調整器が使用する周波数範囲と調整器が使用率をチェックする頻度、使用率の閾値と周波数の設定間隔をそれぞれ設定することができます。

11.2.2 関連するファイルとディレクトリ

お使いのシステムで CPUfreq サブシステムが有効化されていれば (openSUSE では規定で有効化されています)、`/sys/devices/system/cpu/` ディレクトリ以下に関連するファイルやディレクトリが存在するはずです。また、このディレクトリ内の内容を一覧表示すると、`cpu{0..x}` というサブディレクトリが各 CPU に対して存在していて、それ以外にもいくつかのファイルが存在することがわかります。さらに、それぞれのプロセッサのディレクトリ以下には `cpufreq` というサブディレクトリが存在していて、それぞれ CPUfreq に関連するパラメータ設定を行なうことができるようになっています。それらのうちのいくつかは (root で) 書き込みが可能で、それ以外は読み込み専用になっています。お使いのシステムでオンデマンド型調整器や保守型調整器を利用している場合は、`cpufreq` 以下のディレクトリに、調整器ごとのサブディレクトリが存在しています。

注記: プロセッサに対する設定項目について

cpufreq ディレクトリ以下に存在する設定は プロセッサによって異なります。すべてのプロセッサに対して同じポリシーを設定したい場合は、それぞれのプロセッサ用にパラメータを調整する必要があります。なお、現在の設定 (/sys/devices/system/cpu*/cpufreq ディレクトリ内) を手作業で参照したり変更したりする代わりに、cpupower パッケージを利用することをお勧めします。

11.3 電源関係の設定の閲覧／管理／チューニング

電源関係の設定について、閲覧や管理、チューニングを行なうには、下記のコマンドラインツールが利用できます：

cpupower ツールの使用 (132 ページ)

新しく提供されるようになった cpupower ツールは、対象のマシンで利用可能な全ての CPU 電源関係のパラメータについて、ターボ (またはブースト) 状態を含む概要を表示できるように設計されています。これらのツールでは CPUfreq サブシステムや cpuidle サブシステムのほか、周波数制御や負荷状態とは関係のない設定についても、制御を行なうことができます。このような統合型フレームワークの仕組みにより、カーネル関連のパラメータとハードウェアの統計情報の両方にアクセスできるため、性能関連のベンチマークを行なうのに便利な仕組みになっています。また、ターボ状態や負荷状態の依存関係を調べる機能もあります。

powerTOP を利用した電力消費の監視 (135 ページ)

powerTOP は様々な情報源からの情報 (プログラムやデバイスドライバ、カーネル オプションや、プロセッサのスリープ状態からの復帰割り込みについて、その量や発信源の分析情報) を組み合わせ、1 つの画面内に表示することができます。これにより、不用意に高い電源消費について原因を探ったり (たとえば、プロセッサのスリープ状態を頻繁に解除しているプロセスを調べたり) することができるほか、これらを回避するためのシステム設定の最適化についても支援を行います。

11.3.1 cpupower ツールの使用

cpupower パッケージをインストール すると、cpupower で始まるサブコマンドを利用できるようになります。詳しくは `cpupower --help` で表示される ヘルプ、もしくは一般的なマニュアルページ `man cpupower` か、各サブコマンドのマニュアルページ `man cpupower-サブコマンド` をお読みください。

`cpufreq-info` や `cpufreq-set` として使用していたコマンドは、それぞれ `frequency-info` と `frequency-set` のサブコマンドに置き換えることができます。ただし、出力内容が拡張されているほか、文法や動作について若干の差異があることに注意してください：

cpufreq と cpupower の文法の違い*

- コマンドを適用する CPU 番号を指定する場合、いずれのコマンドとも `-c` オプションを使用します。ただしサブコマンド構造により、`-c` を書くべき場所が `cpupower` では異なっています：

```
cpupower -c 4 frequency-info (cpufreq-info -c 4 と同じ意味になります)
```

`cpupower` では `-c` を利用して適用する CPU を指定します。たとえば下記のコマンドでは、それぞれ 1, 2, 3, 5 の各 CPU に設定を適用する意味になります：

```
cpupower -c 1-3,5 frequency-set
```

- `cpufreq*` と `cpupower` を `-c` オプション無しで実行した場合、動作が異なります：

`cpufreq-set` では、自動的に CPU 0 に対する処理として解釈するのに対し、`cpupower frequency-set` では全ての CPU に対する処理と解釈します。また、`cpupower *info` コマンドに何も指定しない場合は、CPU 0 にのみアクセスするのにに対し、`cpufreq-info` は全ての CPU にアクセスします。

11.3.1.1 cpupower を利用した設定の確認

`cpufreq-info` と同じく、`cpupower frequency-info` でも、カーネル内で 使用されている `cpufreq` ドライバの統計情報を表示することができます。これに加えて、`cpupower` では BIOS でターボ機能 (ブースト機能) に対応しているかどうかも表示することができます。何もオプションを指定しないで実行した場合、出力は下記のようになります：

例 11.1 *cpupower frequency-info* の出力例

```
analyzing CPU 0:
  driver: acpi-cpufreq
  CPUs which run at the same hardware frequency: 0 1 2 3
  CPUs which need to have their frequency coordinated by software: 0
  maximum transition latency: 10.0 us.
  hardware limits: 2.00 GHz – 2.83 GHz
  available frequency steps: 2.83 GHz, 2.34 GHz, 2.00 GHz
  available cpufreq governors: conservative, userspace, powersave, ondemand, performance
  current policy: frequency should be within 2.00 GHz and 2.83 GHz.
                    The governor "ondemand" may decide which speed to use
                    within this range.
  current CPU frequency is 2.00 GHz (asserted by call to hardware).
  boost state support:
    Supported: yes
    Active: yes
```

全ての CPU に対する設定値を表示したい場合は、`cpupower -c all frequency-info` と実行してください。

11.3.1.2 *cpupower* を利用したカーネル負荷状態の表示

`idle-info` サブコマンドでは、カーネル内で使用されている `cpuidle` ドライバに関する統計情報を表示することができます。`cpuidle` カーネル フレームワークに対応するアーキテクチャであれば、全てのアーキテクチャで動作します。

例 11.2 *cpupower idle-info* の出力例

```
CPUidle driver: acpi_idle
CPUidle governor: menu

Analyzing CPU 0:
Number of idle states: 3
Available idle states: C1 C2
C1:
Flags/Description: ACPI FFH INTEL MWAIT 0x0
Latency: 1
Usage: 3156464
Duration: 233680359
C2:
Flags/Description: ACPI FFH INTEL MWAIT 0x10
Latency: 1
Usage: 273007117
Duration: 103148860538
```

11.3.1.3 cpupower を利用したカーネルとハードウェア状態の監視

cpupower で最もパワフルな拡張が monitor サブコマンドです。プロセッサの構造や周波数を報告するほか、特定の時間内における負荷状態の統計情報を表示することができます。規定の時間間隔は 1 秒ですが、-i オプションを指定することで変更することもできます。また、本ツールでは個別のプロセッサのスリープ状態と周波数カウンタも実装されています。これらはカーネルの統計情報からの値と、ハードウェアレジスタの値からそれぞれ採取した値です。利用可能なモニタは、お使いのシステムのハードウェアに依存します。利用可能なモニタを表示するには、cpupower monitor -l コマンドを実行してください。個別の監視機能に関する説明は、cpupower-monitor のマニュアル ページをお読みください。

monitor サブコマンドは、特定の負荷に対して性能のベンチマークを実行することができるほか、カーネルの統計情報とハードウェアの統計情報を比較することもできます。

例 11.3 cpupower monitor の出力例

① Mperf				② Idle_Stats			
CPU	C0	Cx	Freq	POLL	C1	C2	C3
0	3.71	96.29	2833	0.00	0.00	0.02	96.32
1	100.0	-0.00	2833	0.00	0.00	0.00	0.00
2	9.06	90.94	1983	0.00	7.69	6.98	76.45
3	7.43	92.57	2039	0.00	2.60	12.62	77.52

- ① Mperf は一定時間内での CPU の平均周波数のほか、ブースト時の周波数を表示します。これに加えて、CPU の動作状態 (C0) と何らかのスリープ状態 (Cx) について、それぞれの時間的な割合も表示されます。既定の採取間隔は 1 秒で、値はハードウェアレジスタから直接読み込まれます。ターボ状態は BIOS で管理されているため、すぐに表示することはできません。ターボ機能を持つ新しいプロセッサの場合、Mperf は特定の CPU に関する周波数情報を確認する唯一の方法となります。
- ② Idle_Stats は cpuidle カーネルサブシステム内の統計情報を表示します。カーネルは負荷状態が切り替わるごとにこれらの値を更新するため、計測の開始と終了の前後では、これらの値が正確にならない場合があります。

上記の例にある (一般的な) 監視機能に加え、アーキテクチャ固有の監視機能も備わっています。詳しい情報は cpupower-monitor のマニュアルページをお読みください。

個別の監視データについて値を比較することで、それらの関係性や依存関係を調べることができるほか、特定の負荷に対して省電力機構が正しく動作しているかどうかを評価することもできます。例 11.3 (134 ページ) の例では、CPU 0 がほぼ無負荷状態 (C_x の値が ほぼ 100%) であるのに、非常に高い周波数で動作しています。それに加えて、CPU 0 と 1 が同じ周波数の値になっているため、それらに依存関係があることを示しています。

11.3.1.4 cpupower を利用した設定変更

cpufreq-set と同様に、root で cpupower frequency-set を実行することで、設定値を変更することができます。これにより調整器が選択する最大と最小の周波数を設定することができるほか、新しい調整器を作成することもできます。また -c オプションを使用すると、設定を適用するプロセッサを選択することもできます。これにより、個別のプロセッサに対していちいち設定することなく、全てのプロセッサに対して一括でポリシーを設定することができます。利用可能なオプションについて、詳しくは cpupower-frequency-set のマニュアルページ、もしくは cpupower frequency-set --help で表示されるヘルプメッセージをお読みください。

11.3.2 powerTOP を利用した電力消費の監視

システムの電源消費量を監視するためにある便利なツールとして、powerTOP というツールもあります。このソフトウェアは、不用意に高く電源を消費している原因を探ることができる (たとえばプロセッサが一時的に休止している状態から、動作状態に戻っている主なプロセスの検出など) ほか、これらを防ぐために適用すべきシステム設定の最適化などを行ないます。なお、powertop パッケージは Intel および AMD の両プロセッサに対応しています。

powerTOP は様々な情報源からの情報を組み合わせて 1 画面に表示します (たとえば プログラムやデバイスドライバの分析、カーネルオプション、一時休止状態から動作状態に戻っているプロセスなど)。例 11.4「powerTOP の出力例」(135 ページ) では、利用可能な情報分類を示しています:

例 11.4 powerTOP の出力例

Cn	Avg	residency	P-states	(frequencies)
①	②	③	④	⑤
C0 (cpu running)		(11.6%)	2.00 Ghz	0.1%
polling	0.0ms	(0.0%)	2.00 Ghz	0.0%
C1	4.4ms	(57.3%)	1.87 Ghz	0.0%

C2 10.0ms (31.1%) 1064 Mhz 99.9%

Wakeups-from-idle per second : 11.2 interval: 5.0s ⑥
no ACPI power usage estimate available ⑦

Top causes for wakeups: ⑧

```
96.2% (826.0)              <interrupt> : extra timer interrupt
0.9% ( 8.0)              <kernel core> : usb_hcd_poll_rh_status (rh_timer_func)
0.3% ( 2.4)              <interrupt> : megasas
0.2% ( 2.0)              <kernel core> : clocksource_watchdog (clocksource_watchdog)
0.2% ( 1.6)              <interrupt> : eth1-TxRx-0
0.1% ( 1.0)              <interrupt> : eth1-TxRx-4
```

[...]

Suggestion: ⑨ Enable SATA ALPM link power management via:
echo min_power > /sys/class/scsi_host/host0/link_power_management_policy
or press the S key.

- ① この列には C-ステートの一覧が表示されます。CPU が動作しているときは 0 の状態に、動作していないときは 0 以外の状態になります。利用可能な C-ステートや、CPU がどれだけ深い スリープ状態に移行するのかについては、CPU によって異なります。
- ② この列には、特定の C-ステートであり続けた平均時間が、ミリ秒単位で表示されます。
- ③ この列には、特定の C-ステートであり続けた時間の割合が表示されます。一時休止状態で省電力をうまく活用するには、できるだけ深い (番号の大きい) C-ステートに移行する必要があります。加えて、これらの C-ステートに 長くあり続けるようにすると、より電力を削減することができます。
- ④ この列にはプロセッサの動作周波数と、お使いのシステムに対応するカーネルドライバが表示されます。
- ⑤ この列には、計測の間に CPU コアが異なる動作周波数にあり続けた時間が表示されます。
- ⑥ CPU が一時休止状態から動作状態に移行した頻度が、秒あたりの回数 (割り込み 回数) で表示されます。低い値ほどより良い省電力であることを示しています。powerTOP では、interval (間隔) と呼ばれる更新間隔を -t オプションで調整することができます。既定値は 5 秒です。
- ⑦ powerTOP をラップトップ機で実行している場合、この行には ACPI 情報が表示され、どれだけの電力を消費しているのかとバッテリー容量が枯渇するまでの予測時間が表示されます。サーバの場合、この情報は利用できません。
- ⑧ なぜシステムが必要以上に動作しているのかについて、理由を表示します。powerTOP はデータの採取間隔内で、もっとも CPU を動作状態に戻している項目についてランキング表示を行ないます。
- ⑨ お使いのマシンで、電源消費を改善するための提案が表示されます。

より詳しい情報については、<http://www.lesswatts.org/projects/powertop/>にある powerTOP プロジェクトのページをお読みください。ヒント情報や FAQ なども記載されています。

11.4 特殊なチューニングオプション

下記のセクションでは、設定しておいたほうがよりよいと思われる各種の設定について述べています。

11.4.1 P-ステート向けのチューニングオプション

CPUfreq サブシステムでは、P-ステート向けにいくつかのチューニングオプションが用意されています。異なる調整器への切り替えを行なうことができるほか、使用する動作周波数の最大値と最小値を設定したり、調整器ごとの詳細パラメータを設定したりすることもできます。

システム稼働中に異なる調整器に切り替えたい場合は、`cpupower frequency-set` コマンドに `-g` オプションを付けて実行します。たとえば下記のコマンドを `root` で実行すると、オンデマンド型の調整器を有効にすることができます：

```
cpupower frequency-set -g ondemand
```

再起動やシャットダウンを行なった後も調整器の設定を維持したい場合は、11.5 項「電源管理プロファイルの作成と使用」(139 ページ) で説明している `pm-profiler` をご利用ください。

調整器が使用する動作周波数の最大値と最小値を設定したい場合は、それぞれ `-d` や `-u` オプションを指定して実行します。

`cpupower` や `cpufreq*` で設定できる項目以外にも、調整器のパラメータを手作業で調整することもできます。たとえば プロセッサ使用率における `nice` 値の無視 (137 ページ) などがあります。

手順 11.1 プロセッサ使用率における *nice* 値の無視

オンデマンド型や保守型の調整器を利用している場合、`ignore_nice_load` と呼ばれるパラメータを変更することができます。

各プロセスには nice 値と呼ばれる値が設定されています。この値はカーネル 側で使用される値で、他のプロセスに比べてどれだけのプロセッサ時間が 必要であるのかを示しています。より大きな nice 値 (「nicer」とも呼びます) ほど、プロセスの優先順位は低くなり、プロセスに与えられる 時間が少なくなります。

オンデマンド型や保守型の調整器を利用している場合、ignore_nice_load パラメータを 1 に設定すると、nice 値が 0 以外に設定されているプロセスが、プロセッサ全体の利用率としてカウントされないようになります。逆に ignore_nice_load が 0 (既定値) に設定されている場合は、すべてのプロセスが利用率としてカウントされます。このパラメータは、プロセッサの資源を大きく占有するプロセスを動作させているものの、これらは調整器の判断材料から外したい場合に便利な機能です。

- 1 まずは下記のようにして、設定を変更したい調整器のサブディレクトリに 移動します:

```
cd /sys/devices/system/cpu/cpu0/cpufreq/conservative/
```

- 2 現在の ignore_nice_load の値を確認するには、下記のように実行します:

```
cat ignore_nice_load
```

- 3 1 に変更したい場合は、下記のように実行します:

```
echo 1 > ignore_nice_load
```

ヒント: コアに対する設定の反映について

cpu0 に対して ignore_nice_load を設定した場合は、残りの全てのコアに対して も自動的に同じ値が適用されます。この場合、調整器のパラメータを変更したい プロセッサ全てに対して、上記の作業を繰り返したりする必要はありません。

動的な周波数制御によって引き起こされる性能損失について、もう1 つ存在する 影響パラメータには、サンプリングレート (調整器が現在の CPU 負荷を確認し、プロセッサの周波数を変更する間隔) があります この設定の既定値は BIOS に 依存して決まる値で、できる限り低い値を設定してください。新しいシステムの 場合は、既定で適切な値が設定されていて、手作業で調整する必要はありません。

11.4.2 C-ステート向けのチューニングオプション

規定では openSUSE は適切な C-ステートを使用します。最適化を行なうに あたって唯一調整すべきパラメータは、sched_mc_power_savings のパラメータです。

すべての CPU コアを最小限に使用するよう負荷を分散させる代わりに、カーネルはできるだけ 少ないコアだけを動作させて、それ以外を一時的に休止させるようにすることができます。これにより不要なコアがより高い C-ステートに移行することになり、結果として省電力を実現できるようになっています。ただし、実際の電力消費には 多数の要素が絡み合ってくるものであることに注意してください。これにはたとえば、利用可能なプロセッサ数や、どの C-ステートに対応しているか (特に C3 から C6 までの、より深い休止) などがあります。

`sched_mc_power_savings` が 0 (既定値) に設定されている場合、特別なスケジューリングは行なわれません。1 に設定されている場合、スケジューラはすべてのプロセッサに対して少しずつ負荷を与えるのではなく、できる限り少ない数のプロセッサで処理を行なおうとします。このパラメータを変更するには、下記の手順で行ないます:

手順 11.2 コア上でのプロセスのスケジュール

- 1 コマンドライン上で root に切り替えます。
- 2 `sched_mc_power_savings` の現在地を表示するには、下記のコマンドを入力します:

```
cpupower info -m
```
- 3 `sched_mc_power_savings` を 1 に設定するには、下記のコマンドを入力します:

```
cpupower set -m 1
```

11.5 電源管理プロファイルの作成と使用

openSUSE では、`pm-profiler` と呼ばれるサーバ用のツールが提供されています。これは設定ファイルを利用して特定の電源管理機能を有効にしたり 無効にしたりするスクリプトのインフラストラクチャです。これにより異なる プロファイルを定義し、別々の設定をファイル内に保持できるようになります。新しいプロファイルを作成する際の設定の雛形は、`/usr/share/doc/packages/pm-profiler/config.template` に配置されています。雛形には、プロファイル内で使用可能な多数のパラメータ が書かれているほか、それらの使用方法やその他のドキュメンテーションへの リンクがコメントで書かれています。なお、作成したプロファイルは `/etc/pm-profiler/` 内に保存するほか、システムが起動 する際に適用されるべきプロファイルは、`/etc/pm-profiler.conf` で設定します。

手順 11.3 電源管理プロファイルの作成と切り替え

新しいプロファイルを作成するには、下記の手順で行ないます:

- 1 /etc/pm-profiler/ ディレクトリ内に、プロファイル 名のサブディレクトリを作成します。下記のように実行します:

```
mkdir /etc/pm-profiler/testprofile
```

- 2 新しいプロファイル用のファイルを作成するには、まずプロファイルの雛形を新しく作成したディレクトリにコピーします:

```
cp /usr/share/doc/packages/pm-profiler/config.template %  
/etc/pm-profiler/testprofile/config
```

- 3 /etc/pm-profiler/testprofile/config 内の設定を編集し、保存します。なお、不要な変数については削除することもできます。削除した場合は、空の (何も設定されていない) ものとして扱われ、該当する設定項目は何も変更されません。

- 4 次に /etc/pm-profiler.conf を編集します。PM_PROFILER_PROFILE では、システムの起動時に適用されるプロファイルを設定します。何も指定しない場合は、既定のシステムまたはカーネル設定が使用されます。新しく作成したプロファイルを使用するようにするには、下記のようにします:

```
PM_PROFILER_PROFILE="testprofile"
```

ここで入力するプロファイル名は、プロファイルの設定ファイルのパス指定に利用した名前 (/etc/pm-profiler/testprofile/config) と一致していなければなりません。なお、/etc/pm-profiler/testprofile/config 内で指定した NAME と一致する必要はありません。

- 5 プロファイルを有効にするには、下記のように実行します:

```
rcpm-profiler start
```

または

```
/usr/lib/pm-profiler/enable-profile testprofile
```

それぞれのプロファイルの設定ファイルについては、エディタなどで作成したり編集したりしなければなりませんが、プロファイルを切り替える作業については YaST から行なうことができます。電源管理設定を開くには、YaST を起動してシステム > 電源管理を選択するか、もしくは root の状態から、コマンドラインで `yast2 power-management` を実行します。ドロップダウンリストには利用可能なプロファイル

が表示されます。既定のプロファイルは、システムの既定値をそのまま受け入れる意味になります。使用したいプロファイルを選択し、完了を押してください。

11.6 トラブルシューティング

BIOS オプションが有効になっていますか？

C-ステートや P-ステートを使用する場合は、BIOS オプションを確認してください：

- C-ステートを利用するには、CPU C State などの オプションが有効になっていることを確認してください。
- P-ステートや CPUfreq の調整器を利用するには、Processor Performance States などのオプションが 有効になっていることを確認してください。

CPU をアップグレードする際は、BIOS についてもアップグレードを行なう 必要があるかどうかを検討する必要があります。BIOS は新しい CPU を 認識し、利用可能な動作周波数をオペレーティングシステム側に渡す 必要があるためです。

CPUfreq サブシステムは有効になっていますか？

openSUSE では CPUfreq サブシステムは既定で有効に設定されます。サブシステムが有効になっているかどうかを確認するには、お使いのシステムで 下記のパスが存在しているかどうかを確認してください： /sys/devices/system/cpu/cpufreq (お使いのシステムが マルチコアである場合は /sys/devices/system/cpu/cpu*/cpufreq)。cpufreq ディレクトリが存在していれば、サブシステムは 有効に設定されています。

ログファイルに何か記載されていませんか？

まずは CPUfreq サブシステム関連のログ出力がないかどうか、syslog (通常は /var/log/messages) を確認します。ここには 深刻なエラーだけが出力されます。

お使いのマシンで CPUfreq サブシステムについて何らかの問題が発生している場合、追加のデバッグ出力を有効に設定してください。これを行なうには、起動時のパラメータに cpufreq.debug=7 を追加するか、もしくは root で下記の コマンドを実行します：

```
echo 7 > /sys/module/cpufreq/parameters/debug
```

上記を実行すると、状態遷移時に dmesg 内に分析に役立つ詳しい情報が出力されるようになります。ただし上記の手順では 冗長な出力が行なわれるため、本当に問題が発生した場合にのみ実施することをお勧めします。

11.7 さらになる情報

- 電源の使用効率を改善するためのコンポーネントについて、3 つの章から 構成される資料 (英語) が下記の URL で公開されています:
 - *Reduce Linux power consumption, Part 1: The CPUfreq subsystem*: http://www.ibm.com/developerworks/linux/library/l-cpufreq-1/?ca=dgr-lnxw03ReduceLXPWR-P1dth-LX&S_TACT=105AGX59&S_CMP=grlnxw03
 - *Reduce Linux power consumption, Part 2: General and governor-specific settings*: http://www.ibm.com/developerworks/linux/library/l-cpufreq-2/?ca=dgr-lnxw03ReduceLXPWR-P1dth-LX&S_TACT=105AGX59&S_CMP=grlnxw03
 - *Reduce Linux power consumption, Part 3: Tuning results*: http://www.ibm.com/developerworks/linux/library/l-cpufreq-3/?ca=dgr-lnxw03ReduceLXPWR-P1dth-LX&S_TACT=105AGX59&S_CMP=grlnxw03
- LessWatts.org プロジェクトは、Linux システムにおける省電力や電源効率の改善を実現するためのプロジェクトです。プロジェクトの Web ページは <http://www.lesswatts.org/> にあります。有益な FAQ セクションは <http://www.lesswatts.org/documentation/faq/index.php> にあり、便利なヒント集なども掲載されています。なお、CPU レベルでのヒントについては <http://www.lesswatts.org/tips/cpu.php> をお読みください。powerTOP については <http://www.lesswatts.org/projects/powertop/> をお読みください。
- ベースボードマネジメントコントローラ (BMC) の搭載されているプラットフォームの場合、サービスプロセッサ経由で追加の電源管理設定オプションにアクセスすることができます。これらの設定は製造元ごとに異なるもので、このガイドでは説明していません。たとえば *HP ProLiant Server Power Management on SUSE Linux Enterprise Server 11—Integration Note* では、HP 社のプラットフォーム固有の電源管理機能と、Linux カーネルの協調方法について説明が書かれています。この文書は <http://h18004.www1.hp.com/products/servers/technology/whitepapers/os-techwp.html> にあります。

パート V. カーネルのチューニング

複数のカーネルバージョンのインストール

12

openSUSE では、カーネルについて複数のバージョンを同時にインストールすることができます。複数のカーネルをインストールすると、起動用の項目と `initrd` が自動的に作成され、特に手作業で何かを行なう必要はありません。システムを再起動すると、新しく追加したカーネルが起動オプション内に表示されるようになります。

この機能を利用することで、古いカーネルを起動できるように残したまま新しいカーネルをインストールすることができるため、カーネルの更新を行なう際に安全に作業を行なうことができます。この機能を利用する場合は、更新ツール (たとえば YaST オンライン更新や更新アプレット) を使用せず、下記に示す手順で作業を行なってください。

警告: サポート権利について

ご自身でコンパイルしたカーネルや、サードパーティが提供するカーネルをインストールした場合、サポート権利が失われることに注意してください。openSUSE に同梱されているカーネルと、openSUSE 向けに提供されている更新の更新チャンネルからの配布物のみがサポート対象です。

ヒント: お使いのブートローダにおけるカーネル設定の確認

複数のカーネルをインストールしたあとは、お使いのブートローダの設定を確認し、既定の起動項目が正しく設定されているかどうか確認することをお勧めします。詳しくは 項「YaST を利用したブートローダの設定」(第7章 ブートローダ *GRUB*, ↑リファレンス) をお読みください。新しくカーネルをインストールした場合に、既定で追加される行を変更したい場合は、新しいカーネルをインストール

する前に `/etc/sysconfig/bootloader` の設定を変更してください。この件に関する詳細は 項「`/etc/sysconfig/bootloader` ファイル」(第7章 ブートローダー *GRUB*, ↑リファレンス) をお読みください。

12.1 複数バージョン対応の有効化と設定

ソフトウェアパッケージについて、複数のバージョンを同時にインストールできる 機能 (マルチバージョンサポート) は既定では無効に設定されています。これを 有効に設定するには、下記のように行ないます:

- 1 `root` で任意のエディタを利用し、`/etc/zypp/zypp.conf` ファイルを開きます。
- 2 `multiversion` と書かれている行に移動します。すべてのカーネルパッケージに対して、マルチバージョン機能を有効に設定するには、下記の行のコメントアウト文字 (`#`) を外します:

```
# multiversion = provides:multiversion(kernel)
```

- 3 特定のカーネルの派生形 (フレーバー) に限定してマルチバージョン機能を有効に 設定するには、`/etc/zypp/zypp.conf` 内の `multiversion` オプションに対して、下記のようにパッケージ名をカンマ区切りで指定します:

```
multiversion = kernel-default,kernel-default-base,kernel-source
```

- 4 最後にファイルを保存して終了します。

12.1.1 未使用のカーネルに対する自動削除設定

複数バージョンに対応させて新しいカーネルを頻繁にテストしていると、起動メニュー の項目が増えてしまい、わかりにくくなってしまいます。また、`/boot` は通常、容量面では限られた領域に配置されるため、場合によっては `/boot` の容量がいっぱいになってしまう場合もあります。このような場合は、YaST または `zypper` を利用して、手作業で未使用の カーネルを削除することができるほか、`libzypp` を設定して自動的に 未使用のカーネルを削除するように設定することもできます。既定では自動での 削除は行なわれません。

- 1 root で任意のエディタを利用し、`/etc/zypp/zypp.conf` ファイルを開きます。
- 2 `multiversion.kernels` の文字列を検索し、下記のような 行のコメントを外してオプションを有効にします。このオプションはカンマ 区切りの一覧で、それぞれ下記のような値を記入することができます:

2.6.32.12-0.7 : 指定したバージョン番号のカーネルを保持します。

latest: もっとも大きなバージョン番号のカーネルを保持します。

latest-N: N 番目に大きなバージョン番号のカーネルを保持します。

running 実行中のカーネルを保持します。

oldest もっとも小さなバージョン番号 (openSUSE に同梱されている オリジナルのカーネル) を保持します。

oldest+N N 番目に古いバージョン番号のカーネルを保持します。

たとえば下記のように記述します。

```
multiversion.kernels = latest,running
```

最新のカーネルと、現在実行中のカーネルを保持します。これは複数バージョンの機能を全く無効化してしまった場合とは異なり、即時に古いほうのカーネルが削除されたりはせず、**次の再起動後に** 古いほうのカーネルが削除されるようになります。

```
multiversion.kernels = latest,latest-1,running
```

新しいほうから 2 つのバージョンのカーネルと、現在実行中のカーネルを保持します。

```
multiversion.kernels = latest,running,3.0.rc7-test
```

最新のカーネルと現在実行中のカーネル、そして `3.0.rc7-test` というバージョンのカーネルを 保持します。

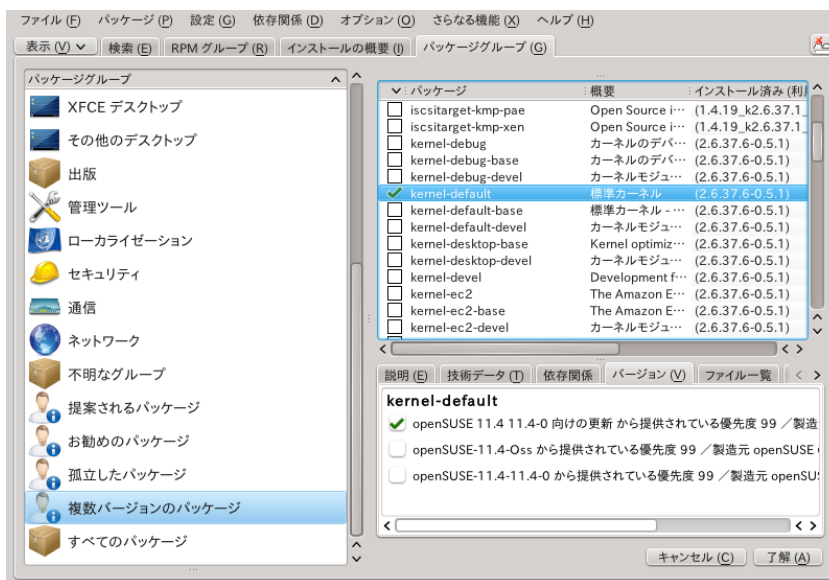
ヒント: 実行中 のカーネルの保持について

特別な場合を除き、実行中 のカーネルは 保持するように設定してください。

12.2 YaST を利用した複数のカーネルバージョンのインストール／削除

- 1 YaST を起動し、ソフトウェア > ソフトウェア管理 を選択してソフトウェア 管理画面を開きます。
- 2 複数のバージョンが提供されているすべてのパッケージを表示するには、表示 > パッケージグループ > 複数バージョンのパッケージ を選択します。

図 12.1 YaST ソフトウェア管理 - 複数バージョン表示



- 3 パッケージを選択し、右下の枠内でバージョン タブを選択します。
- 4 パッケージをインストールするには、それぞれのチェックボックスにチェックを入れます。緑色のチェックマークが表示されます。

既にインストールされたパッケージ (白色のチェックマークが付けられたもの) をアンインストールするには、チェックボックスを選択して赤い X を付けておきます。

- 5 最後に 了解 を押すとインストール作業が始まります。

12.3 zypper を利用した複数のカーネルバージョンのインストール／削除

- 1 利用可能なすべてのカーネルパッケージを一覧表示するには、zypper se -s 'kernel*' コマンドを実行します:

S	名前	種類	バージョン	アーキテクチャ	リポジトリ
v	kernel-default	パッケージ	2.6.32.10-0.4.1	x86_64	Alternative Kernel
i	kernel-default	パッケージ	2.6.32.9-0.5.1	x86_64	(System Packages)
	kernel-default	ソースパッケージ	2.6.32.10-0.4.1	noarch	Alternative Kernel
i	kernel-default	パッケージ	2.6.32.9-0.5.1	x86_64	(System Packages)
...					

- 2 インストール時には、下記のようにして正確なバージョンを指定します:
- ```
zypper in kernel-default-2.6.32.10-0.4.1
```
- 3 カーネルをアンインストールするには、まず zypper se -si 'kernel\*' を実行してインストール済みのすべてのカーネルを一覧表示してから、zypper rm パッケージ名-バージョン のようにしてパッケージを削除します。



## I/O 性能のチューニング

I/O スケジュールはストレージに対して、入出力操作をどのように送信するのかを決めるための仕組みです。openSUSE では様々な I/O アルゴリズムに対応しています。それらのアルゴリズムはエレベータと呼ばれ、それぞれ異なる処理用に用意されています。エレベータはストレージへの要求回数を減らすための手助けになるほか、I/O 要求に優先順位を付けたり、I/O 要求が指定された期限内に処理されるようにしたりすることもできます。

適切な I/O エレベータを選択する際は、処理内容だけでなくハードウェアについても注意しておく必要があります。ATA ディスクが 1 台だけの環境や SSD, RAID アレイやネットワークストレージシステムでは、それぞれ異なるチューニング戦略を立てる必要があります。

### 13.1 I/O スケジューラの切り替え

openSUSE では、システムの起動時に既定の I/O スケジューラが設定されていて、これらはブロックデバイスごとに即時に変更することができます。これにより、システムパーティションを提供するデバイスとデータベースを提供するデバイスなどで、異なるアルゴリズムを設定できるようになっています。

既定では、CFQ (Completely Fair Queuing; 完全公平型キューイング) が使用されます。起動パラメータで変更を行ないたい場合は、下記のように指定します:

```
elevator=SCHEDULER
```

ここで *SCHEDULER* には *cfq*, *noop*, *deadline* のいずれかを指定します。詳しくは 13.2 項「利用可能な I/O エレベータ」(152 ページ) をお読みください。

また、稼働中のシステムで特定のデバイスに対してエレベータを設定するには、下記のように実行します:

```
echo SCHEDULER > /sys/block/DEVICE/queue/scheduler
```

ここで *SCHEDULER* には *cfq*, *noop*, *deadline* のいずれかを指定します。また、*DEVICE* にはブロックデバイス名 (*sda* など) を指定します。

## 13.2 利用可能な I/O エレベータ

openSUSE では、下記のエレベータを使用することができます。それぞれのエレベータにはチューニング可能なパラメータが存在していて、これらは下記のコマンドで設定することができます:

```
echo VALUE > /sys/block/DEVICE/queue/iosched/TUNABLE
```

ここで *DEVICE* にはブロックデバイス名、*TUNABLE* にはパラメータ名称、*VALUE* にはパラメータの値をそれぞれ指定します。

どのエレベータが現在の既定値であるのかを調べるには、下記のコマンドを実行します。現在選択されているエレベータは大括弧で括られて表示されます:

```
jupiter:~ # cat /sys/block/sda/queue/scheduler
noop deadline [cfq]
```

### 13.2.1 CFQ (Completely Fair Queuing; 完全公平型キューイング)

CFQ は公平性を目指したスケジューラで、openSUSE では既定値になっています。このアルゴリズムは各スレッドに対し、I/O リクエストを送信できる時間間隔 (タイムスライス) を付与します。この方法により、各スレッドは I/O スループットを公平に共有することができるようになっています。また、スケジューリングの決定に際して I/O 優先順位を割り当てることもできます (詳しくは `man 1 ionice` をお読みください)。CFQ スケジューラでは、下記のパラメータを設定することができます:

```
/sys/block/<デバイス>/queue/iosched/slice_idle
```

特定のタスクに対して割り当てられた時間 (タイムスライス) 内に処理したい I/O 要求が存在しなくなった場合、I/O スケジューラは I/O の遅延を考慮して、次のスレッドにスケジュールを割り当てるまでにしばらく待機を行いません。大きく遅延しないようなメディア (SSD や多数のディスクが搭載された SAN など)

の場合は、`/sys/block/<デバイス>/queue/iosched/slice_idle` を 0 に設定することで、全体のスループットを高めることができます。

`/sys/block/<デバイス>/queue/iosched/quantum`

このオプションは、デバイスが一度に処理する要求について、その最大数を制限します。既定値は 4 に設定されています。何台かのディスクが搭載されたストレージシステムの場合は、要求を並行処理することから、制限を行なう必要はありません。また、この値を大きくすると性能を改善することができますが、場合によってはストレージ内に多くの要求を貯めこむことになるため、I/O の遅延が大きくなることがあります。この値を変更する場合は、`/sys/block/<デバイス>/queue/iosched/slice_async_rq` (既定値は 2) についても修正を行ない、タイムスライス内に処理する非同期要求 (通常は書き込み要求) について、その最大数を設定しておくことをお勧めします。

`/sys/block/<デバイス>/queue/iosched/low_latency`

I/O の遅延が許されないような負荷を処理する場合は、`/sys/block/<デバイス>/queue/iosched/low_latency` を 1 に設定してください。

## 13.2.2 NOOP

スケジューラに到達した I/O 要求を、単純にそのまま流すだけのものです。複雑な I/O スケジューラを利用する際、何もしない場合と比べて性能が落ちていたりしてないかどうかを確認する際に便利なスケジューラです。

I/O のスケジューリングを自分自身で行なうようなデバイス (たとえば インテリジェント型ストレージ) や、SSD のように機械的な動作を必要としないストレージシステムを利用している場合にも有効です。通常、これらのデバイスには DEADLINE I/O スケジューラが有効ですが、オーバーヘッドが少ないという理由から、NOOP のほうがよりよい性能を発揮できる場合もあります。

## 13.2.3 DEADLINE

DEADLINE は遅延回避を重視する I/O スケジューラです。それぞれの I/O 要求には期限が設定され、通常はセクタ番号を基準にしてキュー (読み込みおよび書き込み) 内の並べ替えが行なわれます。要求が期限切れにならない限り、このような「sector」キューを使用します。期限切れになるものが発生すると、期限切れの要求がなくなるまで「deadline」キューからの処理を実施します。また一般に、このアルゴリズムでは、書き込みよりも読み込みを優先して処理します。

また、この I/O スケジューラは、複数のスレッドが読み書きを行なう種類の処理で、公平性が必要とされない処理の場合に、CFQ よりも優れた性能を提供します。たとえば SAN から変更処理で読み出しを行なうような 場合や、データベース (特に「TCQ」と呼ばれる機能を使用するディスク) などで有効です。なお、DEADLINE スケジューラには、下記のようなチューニングパラメータが存在しています:

```
/sys/block/<デバイス>/queue/iosched/writes_starved
```

書き込み要求を送信できるようになるまでに、どれだけの読み込み要求を送信することができるのかを制御します。たとえば 3 の場合、1 階の書き込み処理を実施する前に、3 回の読み込み処理を実施します。

```
/sys/block/<デバイス>/queue/iosched/read_expire
```

読み込み処理に対する期限をミリ秒単位で指定します。現在の時刻に read\_expire を足した値がその 期限となります。既定値は 500 です。

```
/sys/block/<デバイス>/queue/iosched/write_expire
```

/sys/block/<デバイス>/queue/iosched/read\_expire と同じ仕組みで、こちらは書き込み処理に対する設定です。既定値は 500 です。

## 13.3 I/O バリアのチューニング

多くのファイルシステム (XFS, ext3, ext4, reiserfs) では、fsync やトランザクションコミットの際に書き込みバリアと呼ばれるものをディスクに送信 します。書き込みバリアは書き込みの順序を守らせるための仕組みで (いづらか 性能面への影響があります)、ディスクの書き込みキャッシュを安全に利用できるようにするためのものです。お使いのディスクにバッテリーが搭載されている ような場合は、バリアを無効化することで性能を改善できる場合があります。

書き込みバリアの送信は、マウントオプションに barrier=0 (ext3, ext4, reiserfs の場合) や nobarrier (XFS) を設定することで無効化できます。

---

### 警告

バリアを無効化した場合、電源が失われたような場合に書き込みキャッシュの 内容の書き込みについて、保証がされなくなります。これにより、ファイル システムの破壊やデータ損失が発生する場合がありますにご注意ください。

---



# タスクスケジューラのチューニング

# 14

openSUSE® のように新しいオペレーティングシステムの場合、通常は同時に複数のタスクが動作します。たとえばメールを受信しながら同時にテキストファイルの検索を行ったり、さらに大きなファイルを外付けのハードディスクにコピーしたり、などのことを行なうことができます。これらのようなシンプルな作業であれば、追加のプロセスをシステム内で起動すれば、それらを同時に行なうことができます。それぞれのタスクに対して必要なシステム資源を提供するには、それぞれのタスクに対して資源を割り当てるための Linux カーネル向けツールが必要になります。より正確に書くと、Linux カーネルにはタスクスケジューラと呼ばれるものが用意されています。

本章では、まずプロセス（タスク）のスケジュール処理に関連する最も重要な用語についての説明を行なっています。また、openSUSE で使用されるタスクスケジューラのポリシーに関する情報やスケジュールの処理方法、タスクスケジューラに関する説明などもあわせて記述されています。そのほかにも、関連する情報へのリンクなども示しています。

## 14.1 前置き

Linux カーネルには、動作中のシステムでタスク（プロセス）を管理するための制御方法が用意されています。タスクスケジューラ（プロセススケジューラと呼ばれる場合もあります）はカーネルに含まれるもので、次に動作させるべきタスクを判断する機能を備えています。これは（Linux のような）マルチタスク型のオペレーティングシステムでは中枢となるコンポーネントの 1 つで、複数のタスクを同時に動作させるにあたって、システム資源を最も効率的に働かせるための作業を行ないます。

## 14.1.1 プリエンプション

タスクスケジューラの原理は非常にシンプルなものです。システム内に実行可能なプロセスが存在する場合、常に少なくとも 1 つ以上のプロセスが動作状態になります。システム内にプロセッサ数よりも多い数のプロセスが存在する場合は、全てのプロセスを同時に動作させることはできません。

そのため、プロセスによっては一時的に停止する (サスペンド 状態にする) 必要があることとなります。これにより他のプロセスが動作できる状態になるわけです。スケジューラは、キュー内に存在するプロセスのうち、どれを次に実行すべきかを決定します。

上述のとおり、Linux やその他の Unix 系 OS は マルチタスク 型のオペレーティングシステムです。これは複数の処理を同時に動作させることができる という意味です。Linux では プリエンプティブ マルチタスク と呼ばれる仕組みで、どのプロセスを一時停止させるのかをスケジューラが決定します。このような強制的な停止機能を プリエンプション と呼びます。全ての Unix 系 OS では、当初の段階からプリエンプティブマルチタスクを提供しています。

## 14.1.2 タイムスライス

特定のプロセスが 占有する にあたって、どれだけの時間 占有させるのかは、事前に設定されます。これをプロセスの タイムスライス と呼び、それぞれのプロセスに対して割り当てられるプロセッサ時間の形式で表わします。タイムスライスを割り当てることによって、スケジューラは実行中のシステムに対して 全体制御を行なうことができるほか、特定のプロセスがプロセッサの資源を占有したりしないようにすることができます。

## 14.1.3 プロセスの優先順位

スケジューラでは、プロセスの実行に対して優先順位を判断する仕組みを備えています。プロセスに対する現在の優先順位を計算するため、タスクスケジューラは複雑な アルゴリズム を用いています。結果として、それぞれのプロセスはプロセッサ上で 実行を「許可される」優先順位の値を持つこととなります。

## 14.2 プロセスの分類

プロセスは通常、それらの用途や振る舞いに応じて分類が行なわれます。各分類の境界は 明確に説明できるものではありませんが、一般的に 2 つの条件が分類の際の並び替えに 利用されます。これらの条件はそれぞれ独立しているもので、相互に排他的なものではありません。

プロセスの 1 つめの分類は、*I/O-バウンド* か、もしくは *プロセッサ・バウンド* のどちらであるのかを判断します。

### I/O-バウンド

I/O とはキーボードやマウス、光学ドライブやハードディスクなどのデバイスに対して、入出力することを意味しています。*I/O-バウンドのプロセス* は、多くの時間をデバイスに対する要求の送信や応答の待機に費やしています。これらは頻繁に動作するものではありませんが短い時間で動作が一時停止し、I/O 要求を待機している他のプロセスに対して、その動作を阻害することはありません。

### プロセッサ・バウンド

もう一方の *プロセッサ・バウンド* のタスクではコードの 実行に時間が費やされるもので、通常はスケジューラから優先権が与えられている間 動作し続けるタイプのものです。これらのプロセスは、I/O 要求を待機している プロセスの動作を阻害することはないので、頻度は低いながら長い時間の処理が かかるものです。

もう一つの分類は、それぞれプロセスを *インタラクティブ* (対話型)、*バッチ* (一括処理型)、*リアルタイム* (即時応答型) に分けるやり方です。

- *インタラクティブ* なプロセスは、キーボードやマウス操作 などの入出力要求に多くの時間を費やします。スケジューラはユーザからの要求が あった場合に即時に 応答しなければならず、応答が遅れるとハングアップしたものと みなされてしまいます。この分類での一般的な遅延限界は 100 [ミリ秒] 程度です。オフィス系アプリケーションやテキストエディタ、画像操作プログラムなどが一般的な *インタラクティブ* なプロセスです。
- *バッチ* プロセスはその処理を裏で実施するものであって、応答に対して敏感になる必要はありません。これらはスケジューラでは低い優先順位に 位置づけられます。マルチメディアコンテンツの変換やデータベースの検索エンジン、ログファイルの分析などが一般的な *バッチ* プロセスです。
- *リアルタイム* なプロセスは、低い優先順位を持つプロセス によって実行が阻害されてはならないものです。そのため、スケジューラはそれらに 対する応答性を保

障るようになっていきます。マルチメディアコンテンツの編集 アプリケーションなどが一般的なリアルタイムプロセスです。

## 14.3 O(1) スケジューラ

Linux カーネルバージョン 2.6 では、O(1) と呼ばれる新しいタスクスケジューラが提供されています (詳しくは ランダウの記号 [<http://ja.wikipedia.org/wiki/%E3%83%A9%E3%83%B3%E3%83%80%E3%82%A6%E3%81%AE%E8%A8%98%E5%8F%B7>] をお読みください)。カーネルバージョン 2.6.22 以降では既定のスケジューラとして 使用されています。このタスクスケジューラの主な処理は、システム内でどれだけ多くの プロセスが実行されていたとしても、一定の時間内にタスクをスケジュールすることに あります。

スケジューラではタイムスライスを動的に計算します。ですが、適切なタイムスライスを 計算するのは、あまりにも複雑な処理になってしまいます。長すぎるタイムスライスを 割り当ててしまうとシステムの応答や反応が悪くなってしまいますし、逆に短すぎる タイムスライスを設定するとプロセス間の切り替えが頻発してしまい、それに伴う オーバーヘッドが大きくなってしまいます。そのため、既定のタイムスライスは比較的 小さめ、たとえば 20 [ミリ秒] 程度で、スケジューラはプロセスの優先順位をベースに タイムスライスを決定します。これにより、高い優先順位を持つプロセスに対して、より頻繁かつ長い時間が割り当てられるようになっていきます。

また、プロセスは自身に割り当てられたタイムスライスを一括で使用することはありません。たとえば、150 [ミリ秒] が割り当てられたプロセスであれば、150 [ミリ秒] の間 ずっと実行し続けるわけではなく、30 [ミリ秒] のスケジュールスロットを 5 [回分] に分けて実行することがあります。それほど長いタイムスライスを必要としない対話的な 処理の場合、このようなアプローチにすることで、できるだけ反応性を良くすることが できるようになっています。

なお、スケジューラはプロセスの優先順位も動的に設定します。スケジューラはプロセスの 振る舞いを監視し、必要であれば優先順位を調整します。たとえば、長い時間にわたって 動作が止まっているプロセスの場合、優先順位を上げることで優先的に実行できるようになります。

# 14.4 Completely Fair Scheduler (完全公平型スケジューラ)

Linux カーネルバージョン 2.6.23 以降では、実行プロセスに対する新しいスケジューリング アプローチが提供されるようになりました。Completely Fair Scheduler (CFS) (完全公平型スケジューラ) と呼ばれるもので、既定の Linux カーネルスケジューラになっています。このスケジューラでは重要な変更と改善が施されているため、本章では このスケジューラについて説明を行なっています。なお、openSUSE の場合、カーネルバージョン 2.6.32 以降 (3.x カーネルも含まれます) に適用されています。スケジューラ環境は複数のパーツから構成されていて、3 種類の主な機能が提供されています：

## モジュール型のスケジューラ設計

スケジューラの中核部分は スケジューリングクラス と呼ばれるもので拡張されています。これらのクラスはモジュールとして利用できるもので、スケジューラ処理のポリシーを表わすものです。

## 完全公平型スケジューラ (CFS)

カーネル 2.6.23 で提供され、2.6.24 で拡張されています。CFS はそれぞれの プロセスに対して、プロセッサ時間を「公平に」共有できるように 処理を行ないます。

## グループスケジューリング

たとえば実行ユーザごとにプロセスをグループ化し、CFS はそれらのグループに対して同じプロセッサ時間が割り当てられるように処理を行ないます。

上記のような構成から、CFS はサーバ用途とデスクトップ用途の両方に対して 最適なスケジューリングを行なうことができるようになっています。

## 14.4.1 CFS の動作原理

CFS はそれぞれの処理に対して公平なアプローチを保証しようとします。タスク スケジューリングに対して最もバランスの取れたやり方を検出するため、赤黒木と呼ばれるコンセプトを利用しています。赤黒木は項目を挿入したり削除したりすることのできるデータ検索用の木構造で、挿入や削除が行なわれてもバランスを崩さない仕組みです。赤黒木について、詳しくは 赤黒木 [<http://ja.wikipedia.org/wiki/%E8%B5%A4%E9%BB%92%E6%9C%A8>] にある wiki ページをお読みください。

処理が **実行キュー** (次に実行すべきプロセスの予定表) 内に投入されると、スケジューラはまず現在時刻を記録します。プロセスがプロセッサ 時間を得られるまで待機している間、そのプロセスの「待機」値が 加算されます。加算される値は、その時点で実行キュー内に存在する処理の合計数と、プロセスの優先順位から計算される値になります。プロセッサがあるプロセスを 実行すると、その「待機」値は減算されます。待機値が一定の レベルを下回ると、そのプロセスはスケジューラによって待機状態になり、その他の プロセスがプロセッサを優先的に利用できるようになります。このようなアルゴリズム により、CFS は「待機」値が常にゼロという理想的な状態になる ようにします。

## 14.4.2 プロセスのグループ化

Linux カーネルバージョン 2.6.24 以降では、CFS をタスク (プロセス) 単体だけでなくユーザやグループに対しても公平性を設定できるようになりました。タスクはユーザやグループによってグループ化され、CFS はそれぞれのタスクに 対して公平になるように調整するのではなく、ユーザやグループに対して公平になる ように調整させることができます。なお、スケジューラはユーザやグループ内の 各タスクに対しても公平になるように調整します。

タスクはそれぞれ下記に示す方法のいずれかでグループ化することができます:

- ユーザ ID ごと
- カーネルのコントロールグループごと

カーネルのスケジューラがタスクをグループ化する方法は、カーネルのコンパイル時 に設定するオプション `CONFIG_FAIR_USER_SCHED` および `CONFIG_FAIR_CGROUP_SCHED` によって設定されます。openSUSE® 12.2 での既定の設定はコントロールグループ (後者) で、これにより必要なグループを作成できるようになっています。詳しくは 第10章 **カーネルのコントロールグループ** (117 ページ) をお読みください。

## 14.4.3 カーネル設定オプション

タスクスケジューラの振る舞いにおける基本的な設定項目は、カーネルの設定 オプションで指定できます。これらのオプションの設定は、カーネルのコンパイル 作業の一部として実施するものです。カーネルのコンパイル作業は複雑な作業で あるため本マニュアルでは説明を行っていません。関連するドキュメント (たとえば <http://ja.opensuse.org/>)

[Configure, Build and Install a Custom Linux Kernel](#) など) をお読みのうえ、作業を行なってください。

---

### 警告: カーネルのコンパイル

openSUSE を同梱のカーネルで動作させない場合、たとえばご自身でコンパイルしたカーネルを利用した場合などは、サポートを受けることができなくなることにご注意ください。

---

## 14.4.4 用語

タスクスケジューリングポリシーに関するドキュメントでは、しばしばいくつかの技術用語を利用して説明しています。下記にその用語のうちのいくつかを説明します:

### 遅延

プロセスが実行されるはずだった時間から、実際にプロセスが実行された時間までの時間間隔。

### 粒度

粒度と遅延の関係は下記の数式で表わされます:

$$gran = ( lat / rtasks ) - ( lat / rtasks / rtasks )$$

ここで *gran* は粒度を、*lat* は遅延を、*rtasks* は実行中のタスク数を表わします。

### 14.4.4.1 スケジュールポリシー

Linux カーネルでは、下記のようなスケジュールポリシーに対応しています:

#### SCHED\_FIFO

特別なアプリケーションに対して適用されるもので、特に時間面に制約のあるアプリケーションに使用します。これは FIFO (First In-First Out) と呼ばれるスケジュール用アルゴリズムを使用します。

#### SCHED\_BATCH

CPU を主に使用する処理向けのスケジュールポリシーです。

#### SCHED\_IDLE

とても優先度の低い処理向けに用意されたスケジュールポリシーです。

## SCHED\_OTHER

ほとんどのプロセスに適用される、既定の Linux 時間共有型スケジューリング ポリシーです。

## SCHED\_RR

SCHED\_FIFO に似た仕組みですが、ラウンドロビン型のスケジューリング アルゴリズムを使用するスケジューリングポリシーです。

# 14.4.5 chrt を利用したリアルタイム属性の変更

chrt は、実行中のプロセスに対してリアルタイム スケジューリングの設定や設定値の取得を行なうためのコマンドであるほか、指定した属性でコマンドを実行することもできるコマンドです。スケジューリング ポリシーの取得のほか、プロセスの優先度を取得することもできます。

下記では、PID が 16244 である場合の例を示しています。

既存の処理に対して、リアルタイム属性を *取得* するには、下記のように実行します：

```
saturn.example.com:~ # chrt -p 16244
pid 16244's current scheduling policy: SCHED_OTHER
pid 16244's current scheduling priority: 0
```

プロセスに対して新しいスケジューリングポリシーを設定する前に、それぞれのスケジューリングアルゴリズムに対して設定できる優先度について、その最大値と最小値を取得しておきます：

```
saturn.example.com:~ # chrt -m
SCHED_OTHER min/max priority : 0/0
SCHED_FIFO min/max priority : 1/99
SCHED_RR min/max priority : 1/99
SCHED_BATCH min/max priority : 0/0
SCHED_IDLE min/max priority : 0/0
```

上記の例では、SCHED\_OTHER, SCHED\_BATCH, SCHED\_IDLE の各ポリシーはそれぞれ優先度 0 のみが許可されています。SCHED\_FIFO と SCHED\_RR は優先度が 1 から 99 までの範囲で設定できることを示しています。

SCHED\_BATCH スケジューリングポリシーを設定するには、下記のように実行します：

```
saturn.example.com:~ # chrt -b -p 0 16244
saturn.example.com:~ # chrt -p 16244
pid 16244's current scheduling policy: SCHED_BATCH
```



```
pid 16244's current scheduling priority: 0
```

chrt について、詳しくはマニュアルページ (man 1 chrt) をお読みください。

## 14.4.6 sysctl を利用した実行時のチューニング

実行時にカーネルのパラメータを調査したり変更したりすることのできる sysctl インターフェイスを利用することで、タスクスケジューラの動作を既定値から変更することができます。sysctl の書式は単純なものですが、下記に示すコマンドはいずれも root で実行しなければなりません。

カーネル変数の値を読み出すには、下記のように入力します:

```
sysctl variable
```

値を設定するには、下記のように入力します:

```
sysctl variable=value
```

スケジューラ関連の sysctl 変数の一覧を読み出すには、下記のように入力します:

```
sysctl -A | grep "sched" | grep -v "domain"

saturn.example.com:~ # sysctl -A | grep "sched" | grep -v "domain"
kernel.sched_child_runs_first = 0
kernel.sched_min_granularity_ns = 1000000
kernel.sched_latency_ns = 5000000
kernel.sched_wakeup_granularity_ns = 1000000
kernel.sched_shares_ratelimit = 250000
kernel.sched_tunable_scaling = 1
kernel.sched_shares_thresh = 4
kernel.sched_features = 15834238
kernel.sched_migration_cost = 500000
kernel.sched_nr_migrate = 32
kernel.sched_time_avg = 1000
kernel.sched_rt_period_us = 1000000
kernel.sched_rt_runtime_us = 950000
kernel.sched_compat_yield = 0
```

なお、「\_ns」や「\_us」で終わる変数は、それぞれナノ秒やマイクロ秒を単位とした値を示しています。

また下記の一覧では、タスクスケジューラの `sysctl` 変数のうち、チューニングに重要なものと、その説明を示しています (いずれも `/proc/sys/kernel/` 内に存在しています):

#### `sched_child_runs_first`

新しく `fork()` された子プロセスが、親プロセスが実行状態に戻る前に実行が開始されるようになります。このパラメータを 1 に設定すると、`fork` された後すぐに子プロセスが動作をはじめるとなようなアプリケーションには、有益な設定になります。たとえば、`make -j<CPU 数>` のようなコマンドラインの場合、`sched_child_runs_first` を無効に設定したほうがよりよい性能を発揮します。既定値は 0 です。

#### `sched_compat_yield`

古い 0(1) スケジューラの動作を有効にし、より積極的な `yield` 動作を行なうようになります。動機 (synchronization) を多用する Java アプリケーションの場合、この値を 1 に設定するとよりよい性能を発揮することができます。ただし、この値は性能面に何らかの問題を抱えた場合にのみ設定してください。既定値は 0 です。

なお、`sched_yield()` システムコールの動作に依存するアプリケーションに対して、その性能を改善したい場合も 1 に設定するとよいでしょう。

#### `sched_migration_cost`

タスクの移行判断の際、「キャッシュホット」であると判断されるまでの時間間隔を、最後の実行からの経過時間で設定します。「ホットな」タスクは移行されにくいので、この値を増加させることでタスク移行を少なくすることができます。既定値は 500000 [ナノ秒] です。

実行すべきプロセスが存在するのに CPU の空き時間が期待よりも多い場合は、この値を小さくしてみるとよいでしょう。逆にタスクが CPU 間やノード間を行き来してしまうような場合は、大きくしてみるとよいでしょう。

#### `sched_latency_ns`

CPU を主に使用するタスクに対して、占有権を得るまでの遅延時間について、その目標値を設定します。この値を増加させると、CPU がタスクに割り当てるタイムスライスが大きくなります。タスクのタイムスライスは、スケジュールされた時間単位を公平に共有するように配分を行ないます:

タイムスライス = スケジュール間隔 \* (タスクの重さ / 実行キュー内にあるタスクの重さの合計値)

タスクの重さは、タスクの nice 値とスケジューリングポリシーに依存して決まります。SCHED\_OTHER におけるタスクの重さの最小値は nice 値が 19 のときで、15 です。タスクの重さの最大値は nice 値が -20 のときで、88761 になります。

タイムスライスは負荷が増加すると小さくなります。動作しているタスクの数が  $\text{sched\_latency\_ns} / \text{sched\_min\_granularity\_ns}$  を超えると、タイムスライスは (動作しているタスク数) \*  $\text{sched\_min\_granularity\_ns}$  に設定されます。上記より少ない場合、タイムスライスは  $\text{sched\_latency\_ns}$  に設定されます。

また、この値は休止中のタスクが動作中として扱われ、計算に含むようにする際の最大時間間隔をも意味しています。この値を増加させると、動作中のタスクが占有権を得るまでにかかる時間が増えます。そのため、CPU を主に使用するタスクの場合は、スケジューラによる遅延が増大することになります。既定値は 20000000 [ナノ秒] です。

#### `sched_min_granularity_ns`

CPU を主に使用するタスクに対して、占有権を得る際の粒度の最小値を設定します。詳しくは `sched_latency_ns` をお読みください。既定値は 4000000 [ナノ秒] です。

#### `sched_wakeup_granularity_ns`

タスク起動時の占有権の粒度を設定します。この値を増加させると起動時の占有性が低くなり、主に計算処理を行なうタスクに対してそれらへの妨害が少なくなります。また、この値を低く設定すると起動時の遅延が少なくなり、遅延させたくないタスクに対するスループットを向上させることができます。特に短い効率サイクルの負荷を持つようなタスクと、CPU を多く使用するタスクを同時に動かさなければならない場合に利用します。既定値は 5000000 [ナノ秒] です。

---

### 警告

この値を `sched_latency_ns` の半分よりも大きく設定すると、起動時の占有性がゼロになり、短い効率サイクルのタスクが CPU を占有するタスクに打ち勝てなくなってしまうです。

---

#### `sched_rt_period_us`

リアルタイムなタスクの帯域強制的割り当て時間を設定します。既定値は 1000000 [マイクロ秒] です。

#### `sched_rt_runtime_us`

リアルタイムなタスクに対して、`sched_rt_period_us` の期間に割り当てる分量を設定します。-1 を設定すると、リアルタイムなタスクに対する帯域強制

を行なわなくなります。既定では、リアルタイムなタスクは 95 [%CPU毎秒] を消費するものとしているため、残りの 5 [%CPU毎秒] または 0.05 [秒] が SCHED\_OTHER に設定されたタスクに割り当てられます。

#### `sched_features`

特定のデバッグ機能の目的で情報提供を行なうためのものです。

#### `sched_stat_granularity_ns`

タスクスケジューラの統計に対する粒度を設定します。

#### `sched_nr_migrate`

移行のソフトウェア割り込み (softirq) の期間に、プロセッサ間を移動できるタスク数を制御します。SCHED\_OTHER のポリシーで多数のタスクが作成される環境では、それらはすべて同じプロセッサで動作し続けます。既定値は 32 です。この値を増やすと、リアルタイムなタスクに対する遅延が増大する一方、SCHED\_OTHER のスレッドに対して性能向上を行なうことができます。

## 14.4.7 デバッグ用インターフェイスとスケジューラ統計情報

CFS では新しいデバッグインターフェイスが提供されているほか、システム動作中の統計情報を提供する機能も備わっています。関連するファイルは /proc ファイルシステム内に配置され、これらは単純に `cat` や `less` のコマンドで調べることができるようになっています。/proc ディレクトリ内に存在する関連ファイルについて、下記に説明を列挙します：

#### `/proc/sched_debug`

すべてのチューニング可能な変数に対して、現在の値を表示します (詳しくは 14.4.6 項「`sysctl` を利用した実行時のチューニング」(163 ページ) を参照)。これらはタスクスケジューラの動作に影響があるもののほか、CFS の統計情報や利用可能なすべてのプロセッサに対する実行キュー情報が含まれています。

```
saturn.example.com:~ # less /proc/sched_debug
Sched Debug Version: v0.09, 2.6.32.8-0.3-default #1
now at 2413026096.408222 msecs
.jiffies : 4898148820
.sysctl_sched_latency : 5.000000
.sysctl_sched_min_granularity : 1.000000
.sysctl_sched_wakeup_granularity : 1.000000
.sysctl_sched_child_runs_first : 0.000000
```

```

.sysctl_sched_features : 15834238
.sysctl_sched_tunable_scaling : 1 (logarithmic)

cpu#0, 1864.411 MHz
.nr_running : 1
.load : 1024
.nr_switches : 37539000
.nr_load_updates : 22950725
[...]
cfs_rq[0]:/
.exec_clock : 52940326.803842
.MIN_vruntime : 0.000001
.min_vruntime : 54410632.307072
.max_vruntime : 0.000001
[...]
rt_rq[0]:/
.rt_nr_running : 0
.rt_throttled : 0
.rt_time : 0.000000
.rt_runtime : 950.000000

runnable tasks:
task PID tree-key switches prio exec-runtime sum-exec sum-sleep

R cat 16884 54410632.307072 0 120 54410632.307072 13.836804 0.000000

```

## /proc/schedstat

現在の実行キュー関連する統計情報を表示します。SMP システムにおけるドメイン 固有の統計情報についても、接続されているプロセッサごとに表示が行なわれます。出力形式は読みやすいものとはいえないものであるため、詳しくは /usr/src/linux/Documentation/scheduler/sched-stats.txt をお読みください。

## /proc/*PID*/sched

プロセス ID が *PID* であるプロセスに対して、そのプロセスに関するスケジュール情報を表示します。

```

saturn.example.com:~ # cat /proc/`pidof nautilus`/sched
nautilus (4009, #threads: 1)

se.exec_start : 2419575150.560531
se.vruntime : 54549795.870151
se.sum_exec_runtime : 4867855.829415
se.avg_overlap : 0.401317
se.avg_wakeup : 3.247651
se.avg_running : 0.323432
se.wait_start : 0.000000
se.sleep_start : 2419575150.560531
[...]
nr_voluntary_switches : 938552
nr_involuntary_switches : 71872

```

|                |   |      |
|----------------|---|------|
| se.load.weight | : | 1024 |
| policy         | : | 0    |
| prio           | : | 120  |
| clock-delta    | : | 109  |

## 14.5 さらになる情報

Linux のタスクスケジューリングについて、凝縮された知識をご希望の場合は、いくつかの情報源をあたってみることをお勧めします。それらのうちのいくつかを 下記に紹介します:

- タスクスケジューラシステムのシステムコールに関する説明。関連するマニュアルページ (たとえば `man 2 sched_setaffinity`) をお読みください。
- スケジューリングに関する一般的な情報は、スケジューリング [<http://ja.wikipedia.org/wiki/%E3%82%B9%E3%82%B1%E3%82%B8%E3%83%A5%E3%83%BC%E3%83%AA%E3%83%B3%E3%82%B0>] の Web ページをお読みください。
- Linux におけるタスクスケジューリングに関する一般的な情報は、Linux スケジューラの内側 [<https://www.ibm.com/developerworks/jp/linux/library/l-scheduler/>] をお読みください。
- 完全公平型スケジューラに関する固有の情報は、Completely Fair Scheduler によるマルチプロセッシング [<https://www.ibm.com/developerworks/jp/linux/library/l-cfs/>] をお読みください。
- 完全公平型スケジューラをチューニングする場合の固有の情報は、Tuning the Linux Kernel's Completely Fair Scheduler [<http://www.hotaboutlinux.com/2010/01/tuning-the-linux-kernels-completely-fair-scheduler/>] (英語) をお読みください。
- Linux のスケジューラポリシーやアルゴリズムに関する有益な講義は、<http://www.inf.fu-berlin.de/lehre/SS01/OS/Lectures/Lecture08.pdf> (英語) から参照できます。
- Linux のプロセススケジューリングにおける、わかりやすい概要は、Robert Love 氏が書かれた *Linux Kernel Development* (ISBN-10: 0-672-32512-8) (英語) がよいでしょう。<http://www.informit.com/articles/article.aspx?p=101760> を参照してください。

- Linux カーネルの内部に関する広範囲の概要を知りたい場合は、Daniel P. Bovet 氏と Marco Cesati 氏が書かれた *詳解 Linuxカーネル* (原文: ISBN 978-0-596-00565-8) (日本語訳: ISBN 978-4873113135) をお読みください。
- タスクスケジューラに関する技術情報は、`/usr/src/linux/Documentation/scheduler` ファイル内に 記述されています。





# メモリ管理サブシステムのチューニング

# 15

カーネルにおけるメモリ管理の動作を理解したりチューニングしたりするには、まずメモリ管理機能の動作概要を理解し、他のサブシステムとどのように協調動作するのかを知っておくことが重要です。

メモリ管理サブシステムは仮想メモリマネージャとも呼ばれ、以降の記述では「VM」と略されます。VM はカーネル全体とユーザプログラム が利用する物理メモリ (RAM) の割り当てを管理する役割を持っています。また、ユーザプロセスに対して仮想メモリ環境の提供 (Linux 拡張機能付きの POSIX API 経由) も行なっています。それ以外にも、VM はメモリが不足したような場合に、キャッシュメモリの解放や「匿名」メモリのスワップアウトなどを 利用することで、メモリの空き容量を増やす作業も行ないます。

VM の調査やチューニングを行なう際に最もよく知っておくべきことは、それらのキャッシュがどのように管理されているのかです。VM キャッシュについて目指すべきゴールは、スワップやファイルシステム (ネットワークファイルシステム を含む) 操作で発生する I/O を最小化することにあります。これは I/O そのものを回避することでも実現できますし、最適な方法で I/O を送信することでも実現 できます。

空きメモリは、必要であればそれらのキャッシュとして使用されます。キャッシュや匿名メモリ用にさらなるメモリが利用できる場合、さらにキャッシュやスワップを 効率的に行ないます。しかしながら、メモリが不足すると、キャッシュは切り詰められ たり、メモリがスワップアウトされたりします。

処理内容にもよりますが、性能を向上するための対策として第一に、搭載するメモリ量を 増やすという対策があります。これによりキャッシュの削減やスワップアウトの頻度を 減らすことになり、性能を向上させることができます。次にやるべきこととして

は、カーネルのパラメータを変更することでキャッシュ方法を変更するという手段もあります。

最後に、処理内容そのものを調査してチューニングする必要があります。お使いのアプリケーション側で、動作するプロセス数やスレッド数が増えるような場合、それぞれのプロセスが別々にメモリ管理されるとすると、VM キャッシュの効率は落ちてしまうほか、メモリのオーバーヘッドも増えてしまいます。また、アプリケーションが自分自身でバッファやキャッシュを割り当てる場合、そのキャッシュを大きくすることは、VM キャッシュとして利用可能なメモリを減らすことにもなってしまいます。ただし、プロセス数やスレッド数を増やすことで I/O の多重度やパイプラインを増やすことができるほか、マルチコア環境でよりよい性能を発揮する場合があります。つまり、ベストな結果を出すには実験が欠かせないことになります。

## 15.1 メモリ使用方法

一般的には、メモリの割り当ては「pinned」(固定) (「unreclaimable」(埋め立て不可能) と呼ばれる場合もあります), 「reclaimable」(埋め立て可能), 「swappable」(スワップ可能) の 3 種類に分類することができます。

### 15.1.1 匿名メモリ

匿名メモリは一般に、プログラムのヒープメモリやスタックメモリ (たとえば `>malloc()` など確保したメモリ) のことを指します。これは `mlock` が設定されている場合や、利用可能なスワップ領域が存在しないような場合を除いて埋め立て可能 (reclaimable) なメモリです。また匿名メモリは、埋め立て可能な状態になる前に、スワップに書かれなければなりません。なお、スワップの入出力 (スワップイン／スワップアウトの両方) は、割り当てとアクセスパターンの違いにより、ページキャッシュの I/O よりも効率が落ちる傾向にあります。

### 15.1.2 ページキャッシュ

ファイルデータに対するキャッシュのことを指します。ファイルがディスクやネットワークから読み出されると、その内容はページキャッシュ内に保管されます。内容がページキャッシュ内で最新の状態にある場合、ディスクやネットワークに対するアクセスは不要になります。tmpfs や共有メモリのセグメントはページキャッシュに有利に働きます。

ファイルに書き込みが行なわれると、ディスクやネットワークに書き込まれる 前に新しいデータがページキャッシュ内に保管されます (つまりライトバック キャッシュになります)。まだ書き込まれていないデータがページ内に存在する 場合、それは「汚れた」(dirty) 状態であると表現します。汚れた状態として分類されていないページは「クリーン」(clean) な状態であると表現します。クリーンなページキャッシュのページは、メモリが 不足した場合、単純にそれらを解放することで埋め立てが可能です。汚れたページ の場合は、埋め立てが行なわれる前にまずクリーンな状態にしなければなりません。

### 15.1.3 バッファキャッシュ

これはブロックデバイス (たとえば /dev/sda など) に対するページキャッシュ です。ファイルシステムの場合、ディスク上にある inode テーブルやアロケーション ビットマップなど、「メタデータ」構造にアクセスする際、バッファ キャッシュを利用します。バッファキャッシュはページキャッシュと同様に 埋め立てることができます。

### 15.1.4 バッファヘッド

バッファヘッドは小規模な補助構造体で、ページキャッシュアクセス上に割り当てられるものです。一般に、これらはページキャッシュやバッファキャッシュの ページがクリーンである場合、簡単に埋め立てることができます。

### 15.1.5 ライトバック

アプリケーションがファイルに対して書き込みを行なうと、ページキャッシュ (および バッファキャッシュ) は汚れた (dirty) 状態になります。ページが一定 時間汚れた状態であり続けた場合や、汚れたメモリ量が RAM 内の一定の割合を 超えた場合は、カーネルがライトバック処理を動作させます。処理のスレッドは 裏側書き込み処理を行なうため、アプリケーションはそのまま動作し続けること ができるようになっています。アプリケーションがページキャッシュを汚して (書き込み処理を行なって) いる速度に追従できない場合、汚れたページ キャッシュは RAM 上で大きな割合を占めるようになってしまいます。このような 場合は、汚れたページキャッシュが閾値を超えないよう、アプリケーション側の 処理が調整されるようになっています。

## 15.1.6 先読み

VM はファイルアクセスのパターンを監視し、場合によっては先読みを行ないます。先読み処理では、まだ要求されていない範囲のデータを、ファイルシステムからページキャッシュ内に読み込みます。これにより、少ない回数で大きな容量の I/O 要求 (より効率的な読み出し) を実現しているほか、I/O 処理を並行処理で行なう (アプリケーションが動作している間に I/O を同時並行で行なう) となっています。

## 15.1.7 VFS キャッシュ

### 15.1.7.1 Inode キャッシュ

これはそれぞれのファイルシステムに対して構成される、メモリ内に存在する inode 構造のキャッシュです。ファイルサイズやパーミッション、所有権や ファイルデータへのポインタなどの属性が含まれています。

### 15.1.7.2 ディレクトリエントリのキャッシュ

これはシステム内に存在する、ディレクトリエントリのメモリ内キャッシュです。ここには名前 (ファイル名) のほか、それが参照する inode や子となるエントリ などが含まれています。このキャッシュは、ディレクトリ構造を参照する際や、名前でファイルにアクセスする際に使用されます。

## 15.2 メモリ使用率の削減

### 15.2.1 malloc (匿名) 使用の削減

openSUSE 12.2 では、アプリケーション側からメモリを割り当てる 際、openSUSE 10 に比べて多くのメモリを割り当てる場合があります。これは glibc がユーザスペースのメモリを 割り当てる際、その既定の動作が変更になっているためです。

詳しくは [http://www.gnu.org/s/libc/manual/html\\_node/Malloc-Tunable-Parameters.html](http://www.gnu.org/s/libc/manual/html_node/Malloc-Tunable-Parameters.html) をお読みになり、これらのパラメータについて調整を行なってください。

また、openSUSE 10 の動作に戻したい場合は、`M_MMAP_THRESHOLD` の値は `128*1024` の値に設定してください。これはアプリケーション側から `mallopt()` を呼び出すことで実施できるほか、アプリケーションの実行前に `MALLOC_MMAP_THRESHOLD` の環境変数を設定しても実現できます。

## 15.2.2 カーネルのメモリオーバーヘッドの削減

埋め立て可能なカーネルメモリ (上述のキャッシュなど) は、メモリが不足した 場合には自動的に削減されます。それ以外の多くのカーネルメモリは簡単には 削減できませんが、作業内容によって確保されるメモリ量が変化します。

そのため、ユーザ側での処理で利用する項目を減らす (具体的にはプロセス数を減らす、ファイルやソケットを開く数を減らすなど) ことで、カーネルのメモリ 使用量を減らすことができます。

## 15.2.3 メモリコントローラ (メモリの制御グループ)

メモリの制御グループ (cgroup) 機能が不要な場合、カーネルのコマンドラインに `cgroup_disable=memory` を追加することで、この機能を無効化することができます。これにより、カーネルのメモリ消費を若干少なくすることができます。

## 15.3 仮想メモリマネージャ (VM) におけるチューニングパラメータ

VM をチューニングする場合、チューニング時に変更した項目が実際の処理に対して完全に反映されるまでには、しばらくの時間が必要であることを理解する必要があります。また、処理が 1 日を通して変化するような場合は、異なる時間帯では異なる振る舞いを見せることにも注意が必要です。さらに言うと、ある特定の条件ではスループットを向上させる変更が、別の条件では逆効果になる場合もあります。

## 15.3.1 再生率

`/proc/sys/vm/swappiness`

この制御は、カーネルが匿名メモリをスワップアウトさせる際、ページキャッシュやその他のキャッシュと比較した場合の頻度を設定するものです。この値を増加させると、スワップ処理が多くなります。既定値は 60 です。

スワップの I/O は、一般にその他の I/O に比べて効率が大きく悪いものです。しかしながら、ページキャッシュ内のページによっては、ほとんど使用されていない匿名メモリよりも頻繁にアクセスされる場合があります。そのため、正しいバランスをとる必要があります。

処理が重くなっている際にスワップの動作が観測される場合、このパラメータを小さく設定すると解決できる場合があります。また、多量の I/O 動作が存在する環境で、システム内のページキャッシュ量が比較的少ない場合、もしくは 休眠状態にあるアプリケーションが多数起動しているような場合、この値を増加させることで性能が改善する場合があります。

ただし、より多くのデータがスワップアウトされると、それらのデータが必要になった場合、スワップからデータを読み出すのに時間を要するようになることに注意してください。

`/proc/sys/vm/vfs_cache_pressure`

この値は VFS キャッシュに使用されているメモリを埋め立てる際のカーネルの動作傾向を、ページキャッシュやスワップとの比率で制御するものです。この値を増加させると、VFS キャッシュの埋め立て比率が増加します。

この値について変更実験を行わず、いつ変更すべきなのかは難しい問題です。slabtop コマンド (procpd パッケージ内) では、カーネル側で使用しているメモリオブジェクトについて、多く使用されているものを表示することができます。VFS キャッシュは "dentry" と "\*\_inode\_cache" というオブジェクト名で表示されます。これらの容量がページキャッシュに比べて大きな容量を消費している場合、この値を増加させるとよいでしょう。また、この増加により、スワップ処理を少なくさせることもできます。既定値は 100 です。

`/proc/sys/vm/min_free_kbytes`

この値は「アトミックな」割り当て (埋め立て処理を待機 できない割り当て) など、特殊な予約領域向けに空けておかなければならない メモリ量を制御します。これはメモリ使用に関して注意深くチューニングしたような場合を除いて、小さく設定すべきではない値です (通常はサーバ アプリケーションよりも組み込み機器向けに有用な設定です)。ログファイル内に「ページ割り当ての失敗

(page allocation failure)」メッセージとスタックトレースが頻繁に出力されるような場合は、これらの エラーが出なくなる程度まで `min_free_kbytes` を大きくすることができます。これらのメッセージがほとんど発生しないような環境では、調整を考える必要はありません。既定値は RAM の容量によって決まります。

## 15.3.2 ライトバック (書き戻し) パラメータ

openSUSE 10 以降のバージョンでは、ライトバック動作における重要な変更点として、ファイルの設定された `mmap()` メモリが即時に汚れたメモリとして扱われる (書き戻す必要があるものとして扱われる) という点があります。以前のバージョンでは `mmap()` が解除された後や、`msync()` のシステムコールが呼び出された際、またはメモリに対して使用量の圧縮が必要となった場合に書き戻されます。

アプリケーションによっては `mmap()` に関する上記の仕様変更が期待通りのものではなく、性能が落ちる場合があります。Berkeley DB (およびこのソフトウェアを使用しているアプリケーション) はこの場合に含まれるソフトウェアの 1 つであり、これによって問題が発生します。この問題に対しては、ライトバック (書き戻し) の割合と回数を増やすことで回避することができます。

```
/proc/sys/vm/dirty_background_ratio
```

これは空き容量と埋め立て可能なメモリの割合を設定するものです。汚れた (ディスクに書き込む必要のある) ページキャッシュがここで設定した割合を超えると、ライトバック (書き戻し) スレッドが起動して汚れたメモリを ディスクに書き戻します。既定値は 10 (%) です。

```
/proc/sys/vm/dirty_ratio
```

上記に似た値です。ただし、この値を超えた場合は、ページキャッシュに 書き込むアプリケーションは一時停止させられ、ライトバック (書き込み) 処理が動作するようになります。既定値は 40 (%) です。

これら 2 つの値は、ページキャッシュのライトバック (書き戻し) 動作に影響を与えるものです。これらの値を増加させると、システム内にさらなる汚れたメモリを 長い時間確保するようになります。システム内にさらなる汚れたメモリが存在することになると、ライトバック処理の I/O を避けることによってスループットを 改善できることが期待され、さらに最適な I/O 処理を行なうことができること になります。ただし、汚れたメモリの量が増えると、メモリを埋め立てなければ ならない場合や、ディスクへの書き戻しの必要が発生した時にデータの整合性 (sync) ポイントで遅延が発生することになります。

## 15.3.3 先読みパラメータ

`/sys/block/<デバイス>/queue/read_ahead_kb`

1 つまたは複数のプロセスがファイルを順に読んでいっている場合、カーネルは プロセス側に対して、データの読み込み待ちが発生しないようにするため、ファイルを前もって読み込んで (先読みして) おきます。実際の先読み量は 動的に計算される仕組みになっていて、どれだけ I/O が順序通りに読み込まれているのかによって決まります。このパラメータでは、単一のファイルに対してカーネルが先読みする最大量を設定します。ファイルの読み込みの際、順序 通りの読み込みが十分早くないことがわかった場合は、この値を増やすことで解決する場合があります。この値を大きくしすぎてしまうと、先読み に 使用したページ キャッシュが使用されるよりも前に埋め立てられてしまう 場合が考えられるほか、不要な I/O によって動作が遅くなってしまう場合があります。既定値は 512 (キロバイト) です。

## 15.3.4 さらなる VM パラメータ

VM のチューニング関連のパラメータについて、完全な一覧は `/usr/src/linux/Documentation/sysctl/vm.txt` ファイル (kernel-source パッケージ内) に書かれています。

## 15.4 不均一型メモリアクセス (Non-Uniform Memory Access (NUMA))

VM についてもう一つ重要になりつつある役割として、適切な NUMA 割り当て 戦略を提供するという役割があります。NUMA とは不均一型メモリアクセスの 略で、今日におけるマルチソケット型のサーバは NUMA に対応しています。NUMA はスワップやキャッシュを性能面で管理する際の第二の考慮事項で、NUMA のメモリ割り当ての改善にあたっては、多数のドキュメントが提供されています。ページの埋め立てに関わる点では、下記の 1 つのパラメータが 存在します:

`/proc/sys/vm/zone_reclaim_mode`

このパラメータは、他のノード上で多くのメモリが空きになっている場合でも、メモリの埋め立て処理をローカルの NUMA ノードで実行するかどうかを制御します。このパラメータは、さらなる NUMA 属性の宣言が行なわれているマシンであれば、自動的に有効に設定されます。



NUMA マシン上で VM キャッシュがすべてのメモリを占有することが許可されない場合、`zone_reclaim_mode` を 1 に設定することができます。0 を設定すると、このような動作を無効にします。

## 15.5 VM 動作の監視

VM の動作を監視するにあたっては、下記のようなシンプルなツールがあります：

1. `vmstat`: このツールでは、VM が現在行なっている処理について、その概要を表示させることができます。詳しくは 2.1.1 項「`vmstat`」(10 ページ) をお読みください。
2. `/proc/meminfo`: このファイルでは、それぞれメモリがどのように使用されているのかを明細に表わしたものを表示できます。詳しくは 2.4.2 項「詳細なメモリ使用量: `/proc/meminfo`」(29 ページ) をお読みください。
3. `slabtop`: このツールでは、カーネルの slab メモリやバッファ ヘッド、ディレクトリ エントリや inode キャッシュ、`ext3_inode_cache` などについて、詳細な情報を表示することができます。このコマンドは、`procps` パッケージに含まれています。



## ネットワークのチューニング

ネットワークサブシステムは比較的複雑な仕組みで、チューニング作業はシステムの 使用シナリオに大きく依存します。それ以外にも、お使いのネットワーク内で利用しているソフトウェアクライアントやハードウェアコンポーネント (スイッチ、ルータ、ゲートウェイなど) など外部要因にも大きく依存します。Linux カーネルでは 信頼性を高め、オーパヘッドの低さやスループットの高さよりも遅延の少なさを 求めた仕組みになっています。その他の設定はセキュリティを危うくする可能性があるものもありますが、これによって性能を改善することもできます。

### 16.1 カーネルのソケットバッファの設定

ネットワーク処理は主に、通信時に使用する TCP/IP プロトコルとソケット インターフェイスを基礎にしています。TCP/IP について、詳しくは 第11章 ネットワークの基礎 (↑リファレンス) をお読みください。Linux カーネルは ソケットバッファを利用してソケットインターフェイス経由でのデータ送受信を 行ないます。カーネル側のソケットバッファについてチューニングの余地があります。

---

#### 重要: TCP の自動チューニングについて

カーネルバージョン 2.6.17 以降では、最大で 4 [MB] までのバッファサイズの完全自動チューニングが行なわれます。これにより、手作業でのチューニングはほとんどの場合、ネットワーク性能を改善することにはつながりません。また、下記に示す変数については何も変更を加えないほうがよりよい結果である場合があるほか、少なくともチューニング作業はより注意深く行なう必要があります。

古いバージョンのカーネルから更新している場合は、これら手作業による TCP チューニングは、自動チューニング機能を利用する観点から取り除いておくべき設定です。

---

/proc ファイルシステム内に存在する特殊なファイル では、カーネルのソケットバッファのサイズと動作を変更することができます。/proc ファイルシステムについて、詳しくは 2.6 項「/proc ファイルシステム」(34 ページ) をお読みください。ネットワーク関連の ファイルは下記のディレクトリにあります:

```
/proc/sys/net/core
/proc/sys/net/ipv4
/proc/sys/net/ipv6
```

一般的な net 以下の変数については、カーネルのドキュメンテーション (linux/Documentation/sysctl/net.txt) で説明されています。ipv4 以下の特殊な変数については、linux/Documentation/networking/ip-sysctl.txt と linux/Documentation/networking/ipv6-sysctl.txt で説明されています。

/proc ファイルシステム内では、たとえばすべての プロトコルに対して最大ソケット受信バッファサイズや最大ソケット送信 バッファサイズを設定することができるほか、TCP プロトコル (ipv4 内) に対してのみ設定することもできます。そのほか、すべてのプロトコルに対して上書き設定を行なうこともできます (core)。

```
/proc/sys/net/ipv4/tcp_moderate_rcvbuf
```

/proc/sys/net/ipv4/tcp\_moderate\_rcvbuf が 1 に設定されている場合、自動チューニングが有効に 設定され、バッファサイズが自動で調整されます。

```
/proc/sys/net/ipv4/tcp_rmem
```

ここで表示される 3 つの値は、それぞれ 1 接続あたりのメモリ受信バッファ サイズの最小値／初期値／最大値を表わします。ここで指定するサイズは、TCP のウインドウサイズではなく実際のメモリ使用量です。

```
/proc/sys/net/ipv4/tcp_wmem
```

tcp\_rmem と同じで、こちらは 1 接続あたりのメモリ 送信バッファサイズの最小値／初期値／最大値を設定します。

```
/proc/sys/net/core/rmem_max
```

アプリケーション側から要求可能な、受信バッファサイズの最大値を設定します。

```
/proc/sys/net/core/wmem_max
```

アプリケーション側から要求可能な、送信バッファサイズの最大値を設定します。

/proc を利用することで、不要な TCP 機能を無効化することもできます (既定ではすべての TCP 機能が有効に設定されています)。たとえば 下記のようなファイルがあります:

```
/proc/sys/net/ipv4/tcp_timestamps
```

TCP タイムスタンプは RFC1323 で定義されているものです。

```
/proc/sys/net/ipv4/tcp_window_scaling
```

TCP のウインドウスケールリングも RFC 1323 で定義されているものです。

```
/proc/sys/net/ipv4/tcp_sack
```

選択確認応答 (SACK) の設定を行ないます。

sysctl コマンドを利用することで、/proc ファイルシステム内にある変数を読み書きすることができます。sysctl は /etc/sysctl.conf ファイルを読み込む仕組みになっているため、システムの再起動後も設定を持続させ 続けることができますという点で、cat (読み込み) や echo (書き込み) を利用するよりは適切な方法です。sysctl を利用すると、すべての変数とその値を読み込むことが 簡単に行なえます。root で下記のようなコマンドを実行すると、TCP 関連の設定を一覧表示することができます:

```
sysctl -a | grep tcp
```

---

### 注記: ネットワーク変数のチューニングによる派生効果について

ネットワーク関連の変数をチューニングすることで、CPU やメモリなど 他のシステム資源に影響がある場合があります。

---

## 16.2 ネットワークのボトルネック検出とネットワークトラフィックの分析

ネットワークチューニングを行なう前に、ネットワーク側のボトルネックや ネットワークのトラフィックパターンについて、あらかじめそれらを区別しておくことが重要です。これらのボトルネックを発見するにあたっては、いくつかの ツールを利用するのがよいでしょう。

ネットワークトラフィックの分析には、netstat, tcpdump, wireshark などのような ツールが便利です。Wireshark はネットワークトラフィックの分析プログラムです。

## 16.3 netfilter

Linux のファイアウォールやマスカレード機能は、netfilter と呼ばれるカーネル モジュールが提供する機能です。これらはルールベースのフレームワークで、高度にカスタマイズすることができます。あるネットワークパケットが特定のルールに該当すると、netfilter はルールで設定されているとおりパケットを受け入れたり 拒否したり、もしくはアドレス変換などの特殊な処理 (「ターゲット」) を行ないます。

netfilter には数多くの設定項目が存在するため、これを考慮しておく必要があります。そのため、多くのルールが設定されている場合、それだけパケットの処理に時間がかかることになります。また、高度な接続追跡機能 (connection tracking) を使用している場合は、これも比較的大きな負荷になるため、ネットワーク処理 全体を遅延させる原因になる場合があります。

詳しくは netfilter/iptables プロジェクトの Web ページ <http://www.netfilter.org> をお読みください。

## 16.4 さらなる情報

- Eduardo Ciliendo, Takechika Kunimasa: 「Linux Performance and Tuning Guidelines」 (2007), esp. sections 1.5, 3.5, and 4.7: <http://www.redbooks.ibm.com/redpapers/abstracts/redp4285.html>
- John Heffner, Matt Mathis: 「Tuning TCP for Linux 2.4 and 2.6」 (2006): <http://www.psc.edu/networking/projects/tcptune/#Linux>

## パート VI. システムダンプの処理





## トレースツール

openSUSE では、お使いのシステムに関する便利な情報を採取するための 様々なツールが提供されています。採取可能な情報としては、たとえばお使いの プログラムにおけるデバッグや問題発見のための情報や、性能を落としている 原因を発見するための情報、もしくは実行中のプロセスに対して使用している システム資源を追跡するための情報などがあります。これらのツールは主に インストールメディア内に含まれていますが、SUSE ソフトウェア開発キット からインストールするもの もあります。

---

### 注記: トレースと性能への影響

プロセスに対してシステムやライブラリ呼び出しのトレース (追跡) を行なった 場合、そのプロセスの性能は大きく落ちます。そのため、トレース関連のツールはデータの収集を行なうときにだけご利用ください。

---

## 17.1 strace を利用したシステムコールのトレース

strace コマンドは、プロセスが利用するシステムコールや 受信するシグナルについて、それらを追跡するためのプログラムです。strace はコマンドを新たに起動してシステムコールを追跡 させることができるほか、既に動作しているコマンドに対して strace を設定することもできます。このコマンドの出力には、システムコールの名称のほか、そこで指定したパラメータ (括弧内) や戻り値などが含まれています。

新しいコマンドを起動してシステムコールの追跡を行なうには、コマンドラインの冒頭に `strace` と入力してから、あとは通常通りのコマンドラインを入力します:

```
tux@mercury:~> strace ls
execve("/bin/ls", ["ls"], [/ * 52 vars */]) = 0
brk(0) = 0x618000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) ¥
= 0x7f9848667000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) ¥
= 0x7f9848666000
access("/etc/ld.so.preload", R_OK) = -1 ENOENT ¥
(No such file or directory)
open("/etc/ld.so.cache", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=200411, ...}) = 0
mmap(NULL, 200411, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f9848635000
close(3) = 0
open("/lib64/librt.so.1", O_RDONLY) = 3
[...]
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) ¥
= 0x7fd780f79000
write(1, "Desktop¥nDocuments¥nbin¥ninst-sys¥n", 31Desktop
Documents
bin
inst-sys
) = 31
close(1) = 0
munmap(0x7fd780f79000, 4096) = 0
close(2) = 0
exit_group(0) = ?
```

既に起動済みのプロセスに対して `strace` を設定するには、`-p` オプションを利用して監視したいプロセスのプロセス ID (PID) を指定する必要があります:

```
tux@mercury:~> strace -p `pidof mysqld`
Process 2868 attached - interrupt to quit
select(15, [13 14], NULL, NULL, NULL) = 1 (in [14])
fcntl(14, F_SETFL, O_RDWR|O_NONBLOCK) = 0
accept(14, {sa_family=AF_FILE, NULL}, [2]) = 31
fcntl(14, F_SETFL, O_RDWR) = 0
getsockname(31, {sa_family=AF_FILE, path="/var/run/mysql"}, [28]) = 0
fcntl(31, F_SETFL, O_RDONLY) = 0
fcntl(31, F_GETFL) = 0x2 (flags O_RDWR)
fcntl(31, F_SETFL, O_RDWR|O_NONBLOCK) = 0
[...]
setsockopt(31, SOL_IP, IP_TOS, [8], 4) = -1 EOPNOTSUPP (Operation ¥
not supported)
clone(child_stack=0x7fd1864801f0, flags=CLONE_VM|CLONE_FS|CLONE_¥
FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_¥
PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tidptr=0x7fd1864809e0, ¥
tls=0x7fd186480910, child_tidptr=0x7fd1864809e0) = 21993
select(15, [13 14], NULL, NULL, NULL
```

```
tux@mercury:~$ strace -e trace=open,write ls -l
open("/etc/ld.so.cache", 0_RDONLY) = 3
open("/lib64/librt.so.1", 0_RDONLY) = 3
open("/lib64/libselinux.so.1", 0_RDONLY) = 3
open("/lib64/libacl.so.1", 0_RDONLY) = 3
open("/lib64/libc.so.6", 0_RDONLY) = 3
open("/lib64/libpthread.so.0", 0_RDONLY) = 3
[...]
open("/usr/lib/locale/cs_CZ.utf8/LC_CTYPE", 0_RDONLY) = 3
open(".", 0_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3
write(1, "addressbook.db.bak\nbin\ncxoffice\n",..., 311) = 311
```

```
tux@mercury:~$ strace -e trace=network -p 26520
Process 26520 attached - interrupt to quit
socket(PF_NETLINK, SOCK_RAW, 0) = 50
bind(50, {sa_family=AF_NETLINK, pid=0, groups=00000000}, 12) = 0
getsockname(50, {sa_family=AF_NETLINK, pid=26520, groups=00000000}, &[12]) = 0
sendto(50, "\24\0\0\0\26\0\1\3\0\315K\0\0\0\0\0\0\0", 20, 0, {sa_family=AF_NETLINK, pid=0, groups=00000000}, 12) = 20
[...]
```

```
tux@mercury:~$ strace -c find /etc -name xorg.conf
/etc/X11/xorg.conf
% time seconds usecs/call calls errors syscall

32.38 0.000181 181 1 execve
22.00 0.000123 0 576 getdents64
19.50 0.000109 0 917 open
19.14 0.000107 0 888 close
 4.11 0.000023 2 10 mprotect
 0.00 0.000000 0 1 write
[...]
 0.00 0.000000 0 1 getrlimit
 0.00 0.000000 0 1 arch_prctl
 0.00 0.000000 0 3 futex
 0.00 0.000000 0 1 set_tid_address
 0.00 0.000000 0 4 fadvise64
 0.00 0.000000 0 1 set_robust_list

100.00 0.000559 3633 33 total
```

特定のプロセスに対して、それらの子プロセスもあわせて監視するようにするには、`-f` を指定します:

```
tux@mercury:~> strace -f rcapache2 status
execve("/usr/sbin/rcapache2", ["rcapache2", "status"], [/* 81 vars */]) = 0
brk(0) = 0x69e000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f3bb553b000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f3bb553a000
[...]
[pid 4823] rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>
[pid 4822] close(4 <unfinished ...>
[pid 4823] <... rt_sigprocmask resumed> NULL, 8) = 0
[pid 4822] <... close resumed>) = 0
[...]
[pid 4825] mprotect(0x7fc42cbbd000, 16384, PROT_READ) = 0
[pid 4825] mprotect(0x60a000, 4096, PROT_READ) = 0
[pid 4825] mprotect(0x7fc42cde4000, 4096, PROT_READ) = 0
[pid 4825] munmap(0x7fc42cda2000, 261953) = 0
[...]
[pid 4830] munmap(0x7fb1fff10000, 261953) = 0
[pid 4830] rt_sigprocmask(SIG_BLOCK, NULL, [], 8) = 0
[pid 4830] open("/dev/tty", O_RDWR|O_NONBLOCK) = 3
[pid 4830] close(3)
[...]
read(255, "\n\n# Inform the caller not only v...", 8192) = 73
rt_sigprocmask(SIG_BLOCK, NULL, [], 8) = 0
rt_sigprocmask(SIG_BLOCK, NULL, [], 8) = 0
exit_group(0)
```

`strace` の出力を分析する必要がある場合や、出力される メッセージが長すぎるため、コンソールで直接確認するのが困難である場合は、`-o` オプションをお使いください。この場合は、プロセスへの 接続や接続解除のメッセージなど、不要な情報は省略されます。これらの メッセージ (通常は標準出力に出力されます) を省略するには、`-q` を指定します。また、各行に対してタイムスタンプを表示したい場合は、`-t` を使用します:

```
tux@mercury:~> strace -t -o strace_sleep.txt sleep 1; more strace_sleep.txt
08:44:06 execve("/bin/sleep", ["sleep", "1"], [/* 81 vars */]) = 0
08:44:06 brk(0) = 0x606000
08:44:06 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f8e78cc5000
[...]
08:44:06 close(3) = 0
08:44:06 nanosleep({1, 0}, NULL) = 0
08:44:07 close(1) = 0
08:44:07 close(2) = 0
08:44:07 exit_group(0) = ?
```

`strace` の動作と出力形式は幅広い範囲でカスタマイズが可能です。詳しくは 関連するマニュアルページ (`man 1 strace`) をお読みください。

## 17.2 ltrace を利用したライブラリ呼び出しのトレース

ltrace はプロセスに対する動的なライブラリ呼び出しを追跡します。strace に似た方法を使用するもので、パラメータの多くも似通っていたり、全く同じ意味であったりします。既定では ltrace は /etc/ltrace.conf または ~/.ltrace.conf にある設定ファイルを使用します。-F 設定ファイル を使用することで、独自の設定ファイルを指定することもできます。

ライブラリの呼び出しに加えて、ltrace では -S オプションでシステムコールを同時に追跡することもできます:

```
tux@mercury:~> ltrace -S -o ltrace_find.txt find /etc -name ¥
xorg.conf; more ltrace_find.txt
SYS_brk(NULL) = 0x00628000
SYS_mmap(0, 4096, 3, 34, 0xffffffff) = 0x7f1327ea1000
SYS_mmap(0, 4096, 3, 34, 0xffffffff) = 0x7f1327ea0000
[...]
fnmatch("xorg.conf", "xorg.conf", 0) = 0
free(0x0062db80) = <void>
__errno_location() = 0x7f1327e5d698
__ctype_get_mb_cur_max(0x7fff25227af0, 8192, 0x62e020, -1, 0) = 6
__ctype_get_mb_cur_max(0x7fff25227af0, 18, 0x7f1327e5d6f0, 0x7fff25227af0,
0x62e031) = 6
__fprintf_chk(0x7f1327821780, 1, 0x420cf7, 0x7fff25227af0, 0x62e031
<unfinished ...>
SYS_fstat(1, 0x7fff25227230) = 0
SYS_mmap(0, 4096, 3, 34, 0xffffffff) = 0x7f1327e72000
SYS_write(1, "/etc/X11/xorg.conf¥n", 19) = 19
[...]
```

-e オプションを利用することで、追跡するイベントの種類を変更することができます。下記の例では、fnmatch と strlen の関数に関連したライブラリ呼び出しについて表示を行なっています:

```
tux@mercury:~> ltrace -e fnmatch,strlen find /etc -name xorg.conf
[...]
fnmatch("xorg.conf", "xorg.conf", 0) = 0
strlen("Xresources") = 10
strlen("Xresources") = 10
strlen("Xresources") = 10
fnmatch("xorg.conf", "Xresources", 0) = 1
strlen("xorg.conf.install") = 17
[...]
```

特定のライブラリに含まれるシンボルだけを表示するには、-l ライブラリへのパス を使用します:

```
tux@mercury:~> ltrace -l /lib64/librt.so.1 sleep 1
```

```
clock_gettime(1, 0x7fff4b5c34d0, 0, 0, 0) = 0
clock_gettime(1, 0x7fff4b5c34c0, 0xffffffff600180, -1, 0) = 0
+++ exited (status 0) +++
```

-n 空白数を指定することで、それぞれ再帰的に呼び出されているライブラリ呼び出しについて、それらをインデント表示させることができます。これによって、出力を読みやすくすることができます。

## 17.3 Valgrind を利用したデバッグとプロファイル

Valgrind はお買いのプログラムに対してデバッグやプロファイル機能を提供するツール集で、高速にエラーもなく動作できる仕組みです。Valgrind はメモリ管理やスレッド処理に関する問題を検出することができるほか、新しいデバッグツールを構築する際のフレームワークとしても利用することができます。

### 17.3.1 インストール

Valgrind は標準の openSUSE 配布物には含まれていません。お使いのシステムにインストールするには SUSE ソフトウェア開発キットを取得し、アドオン製品として追加したあと、

`zypper install valgrind` を実行します。

それ以外にも、SUSE ソフトウェア開発キットのディレクトリツリーを参照して Valgrind パッケージの場所を見つけ、

`rpm -i valgrind-バージョン/アーキテクチャ.rpm` のように実行してもかまいません。

### 17.3.2 サポート対象のアーキテクチャ

Valgrind は下記のアーキテクチャで動作します:

- i386
- x86\_64 (AMD-64)
- ppc

- ppc64
- System z

## 17.3.3 一般的な情報

Valgrind を使用する主な目的は、既存のコンパイル済み実行ファイルを利用して調査するという点にあります。Valgrind を使用するにあたってコンパイルし直したり、プログラムを修正したりする必要はありません。Valgrind は 下記のようにして実行します:

`valgrind` *Valgrind のオプション* プログラム *プログラムのオプション*

Valgrind は複数のツールから構成されていて、それぞれのツールは固有の機能が用意されています。また、この章における情報は、使用するツールに関係なく一般的に 利用可能なものだけを紹介しています。その中でも最も重要な設定オプションは `--tool` で、このオプションは Valgrind に対して実行する ツールを指定するためのものです。このオプションを省略すると、既定で `memcheck` が選択されます。たとえば Valgrind の `memcheck` ツールを `find ~ -name .bashrc` というコマンドラインに対して 実行したい場合は、下記のように入力します:

```
valgrind --tool=memcheck find ~ -name .bashrc
```

また、下記には標準的な Valgrind ツールと、それについての簡単な説明を示します:

`memcheck`

メモリエラーの検出を行ないます。お使いのプログラムが正しく動作するか どうかを確かめることができます。

`cachegrind`

キャッシュの予測をプロファイルします。お使いのプログラムが高速に 動作するかどうかを確かめることができます。

`callgrind`

`cachegrind` に似た動作を行ないますが、キャッシュプロファイル情報を追加で 収集します。

`exp-drd`

スレッドエラーを検出します。複数のスレッドが動作するプログラムについて、正しく動作するかどうかを確かめることができます。

helgrind

もう 1 つのスレッドエラー検出器です。exp-drd に似た機能ですが、問題の分析時に異なる技術を使用します。

massif

ヒープのプロファイルを行ないます。ヒープとは動的に割り当てられるタイプのメモリ領域のことで、これによってお使いのプログラムをより少ないメモリで動作できるように、チューニングすることができます。

lackey

基本的な仕組みを理解するためのサンプルツールです。

## 17.3.4 既定のオプション

Valgrind は起動時にオプションを読み込みます。オプションは下記の 3 種類の場所で指定できます:

1. Valgrind を実行するユーザのホームディレクトリ内にある `.valgrindrc` ファイル。
2. 環境変数 `$VALGRIND_OPTS`。
3. Valgrind を起動した時点でのカレントディレクトリにある `.valgrindrc` ファイル。

これらのパラメータは上記の順序で処理され、後者での設定は前者での設定を上書きすることができるようになっています。特定の Valgrind に対する固有のオプションは、ツール名に続けてコロン (:) を追加し、その後にオプション名を続けます。たとえば cachegrind に対して、プロファイルデータを必ず `/tmp/cachegrind_PID.log` ファイルに書き込ませたい場合は、ホームディレクトリ内の `.valgrindrc` ファイルで、下記のように設定します:

```
--cachegrind:cachegrind-out-file=/tmp/cachegrind_%p.log
```

## 17.3.5 Valgrind の動作概要

Valgrind は、対象のプログラムが起動する前の段階から制御を獲得するようになっています。まずは実行ファイルからデバッグ情報を読み込み、関連する共有ライブラリを読み込みます。実行ファイル内のコードは選択した Valgrind ツールに転送され、ツールはデバッグを行なうためのコードを追加します。最後にコードは Valgrind の中枢部分に戻され、実行が開始されます。



たとえば `memcheck` では、それぞれのメモリアクセス に対してチェックを行なうコードを追加します。これにより、ネイティブな実行 環境に比べるとずっと遅い動作になってしまいます。

Valgrind では、プログラム内の各命令をシミュレートします。そのため、プログラムのコードをチェックするだけではなく、関連するライブラリ (C ライブラリを含む) やグラフィカル環境のライブラリなどについてもチェックを行なうことができます。つまり、Valgrind でエラーを検出する場合は、関連するライブラリ (C, X11, Gtk ライブラリなど) に関するエラーもチェックすることになります。これらのエラーが不要な場合は、Valgrind 側に設定を行なうことで 出力を省略することができます。--gen-suppressions=yes を指定すると、これら省略したものをレポートさせることができます。

なお Valgrind を実行する際は、実際の実行ファイル (マシンコード) に対して 実行してください。たとえばシェルや Perl スクリプトからアプリケーションを起動すると、`/bin/sh` や `/usr/bin/perl` を調査することになってしまい、誤った結果をもたらしてしまいます。このような 場合は `--trace-children=yes` を指定するか、もしくはシェルやスクリプトではなく、実行ファイルを指定して実行 してください。

## 17.3.6 メッセージ

Valgrind の実行中には、詳細なエラー内容や重要なイベントを含むメッセージが随時表示されます。たとえば下記のようになります:

```
tux@mercury:~> valgrind --tool=memcheck find ~ -name .bashrc
[...]
==6558== Conditional jump or move depends on uninitialised value(s)
==6558== at 0x400AE79: _dl_relocate_object (in /lib64/ld-2.11.1.so)
==6558== by 0x4003868: dl_main (in /lib64/ld-2.11.1.so)
[...]
==6558== Conditional jump or move depends on uninitialised value(s)
==6558== at 0x400AE82: _dl_relocate_object (in /lib64/ld-2.11.1.so)
==6558== by 0x4003868: dl_main (in /lib64/ld-2.11.1.so)
[...]
==6558== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
==6558== malloc/free: in use at exit: 2,228 bytes in 8 blocks.
==6558== malloc/free: 235 allocs, 227 frees, 489,675 bytes allocated.
==6558== For counts of detected errors, rerun with: -v
==6558== searching for pointers to 8 not-freed blocks.
==6558== checked 122,584 bytes.
==6558==
==6558== LEAK SUMMARY:
==6558== definitely lost: 0 bytes in 0 blocks.
==6558== possibly lost: 0 bytes in 0 blocks.
==6558== still reachable: 2,228 bytes in 8 blocks.
==6558== suppressed: 0 bytes in 0 blocks.
```

```
==6558== Rerun with --leak-check=full to see details of leaked memory.
```

==6558== として表示されている数値はプロセスの ID 番号 (PID) で、それに続けて Valgrind からのメッセージが表示されます。これによりプログラム自体からの出力と Valgrind からの出力を区別することができるほか、どのプロセスに属するメッセージなのかがわかるようになっています。

Valgrind のメッセージをより詳細に表示したい場合は、`-v` や `-v -v` の各オプションを使用します。

基本的には Valgrind は自分自身のメッセージを 3 箇所に出力することができます:

1. 既定では Valgrind は自分自身のメッセージをファイルディスクリプタ 2 に送信します。これは一般的には標準エラー出力 (stderr) と呼ばれているもので、独自のファイルディスクリプタに送信させたい場合は、`--log-fd=ファイルディスクリプタ番号` オプションを使用します。
2. 2 番目の出力先としては、`--log-file=ファイル名` で指定するファイル出力があります。なお、このオプションではいくつかの変数を使用することができます。たとえば `%p` は、現在プロファイルされているプロセスのプロセス ID (PID) に置き換えることができます。この方式を使用すると、プロセス ID ごとに異なるファイルに出力させることができます。また `%q{env_var}` では、`env_var` の名前を持つ環境変数について、設定されている値に置き換えることができます。

下記の例では、Apache Web サーバが再起動する際にメモリエラーが存在していないかどうかをチェックしています。このとき、子プロセスについてもチェックを行なうものとし、詳細な Valgrind からのメッセージはそれぞれ現在の プロセス ID で示されるファイルに出力しています:

```
tux@mercury:~> valgrind -v --tool=memcheck --trace-children=yes \
--log-file=valgrind_pid_%p.log rcapache2 restart
```

このプロセスはテストシステム内に 52 個のログファイルを作成し、Valgrind を使用しない場合は通常 7 秒程度かかる `rcapache2 restart` コマンドが 10 倍の 75 秒程度かかる環境であるものとします。すると、下記のような状況になります。

```
tux@mercury:~> ls -l valgrind_pid_*log
valgrind_pid_11780.log
valgrind_pid_11782.log
valgrind_pid_11783.log
[...]
valgrind_pid_11860.log
valgrind_pid_11862.log
```

valgrind\_pid\_11863.log

3. Valgrind のメッセージをネットワーク経由で送信したい場合は、それぞれ aa.bb.cc.dd の形式で IP アドレスを、番号 の形式でポート番号を用意し、--log-socket=aa.bb.cc.dd:port\_num のようにして実行します。ポート番号を省略した場合は、1500 が使用されます。

なお、送信先のマシン側には Valgrind のメッセージを受信するため、アプリケーションを起動する必要があります。このアプリケーションとしては、Valgrind 同梱の valgrind-listener を利用するのがよいでしょう。これは指定したポートで接続を待ち受けて、受信した メッセージを標準出力に出力します。

## 17.3.7 エラーメッセージ

Valgrind はすべてのエラーメッセージを記憶し、新しいエラーを検出すると 古いエラーメッセージとの比較が行なわれます。この方法により、Valgrind は エラーメッセージの重複をチェックしています。重複したエラーであることが 判明した場合、それは記憶されるだけで表示はされません。これにより、数多くのエラーメッセージが表示されることによる混乱を防ぐようになっています。

なお、-v オプションを指定することで、Valgrind の実行時 出力の最後に全レポートの概要 (発生回数順) を表示させることができます。ただし、Valgrind は 1,000 種類以上の異なるエラーを検出した場合や、全部で 10,000,000 回以上のエラーを検出した場合は、エラー収集を停止してしまうことに注意してください。この制限を取り払ってすべてのエラーメッセージを表示 させたい場合は、--error-limit=no を指定してください。

また、あるエラーが別のエラーを呼び起こす場合もあります。このような場合は、発生した順序どおりに問題を修正し、プログラムを再度チェックすることをお勧めします。

## 17.4 さらなる情報

- 上記のトレースツールに関連したオプションについて、完全な一覧を読みたい場合は、それぞれのマニュアルページ (man 1 strace, man 1 ltrace, man 1 valgrind) をお読みください。
- Valgrind のより高度な使用方法に関する説明は、Valgrind User Manual [<http://valgrind.org/docs/manual/manual.html>] (英語) をお読みください。

これらのページは Valgrind を高度な機能を使用したり、標準ツールの使用方法や目的を調べたりしたい場合には欠くことのできないものです。

## kexec と kdump

kexec は現在実行中のカーネルとは別のカーネルを起動するためのツールです。これにより、ハードウェアの初期化作業を省略して、より高速なシステムの起動を行なうことができます。またシステムがクラッシュした場合に備え、他のカーネルを起動するための準備としても利用することができます。

### 18.1 概要

kexec を利用することで、ハードウェア側の再起動を利用することなく、実行中のカーネルを入れ替えることができます。このツールは下記のような理由により、便利な仕組みになっています：

- より高速なシステム再起動のため

何らかの理由でシステムを定期的に再起動しなければならない場合、kexec は再起動にかかる時間を削減することができます。

- 信頼できないファームウェアやハードウェアの回避のため

コンピュータのハードウェアは複雑なものになってしまっていて、重大な問題がシステムの起動時に発生してしまう場合があります。このような 場合、即時に信頼のできないハードウェアを取り替える必要がありますが、kexec を利用することで、ハードウェアが既に初期化された、制御可能な 状態でカーネルを起動することができます。これによって、システムの 起動が失敗してしまうリスクを最小限に抑えることができます。

- クラッシュしたカーネルのダンプ保存のため

kexec は物理メモリ内の内容を保持するような仕組みになっています。本番用として使用しているカーネルの動作に何らかの問題が発生した場合、キャプチャ専用のカーネル (予約されたメモリの範囲内で動作する追加のカーネル) が動作して、問題が発生したときの状態を保存することができます。保存されたイメージは、さらなる分析のために利用することができますようになります。

- GRUB や LILO の設定を使用しない起動のため

kexec を利用してカーネルを起動する場合、ブートローダの手順を飛ばして動作します。これにより、通常の起動処理が、ブートローダ側の設定ミスによって失敗してしまうようなリスクを防ぐことができます。kexec では、既存のブートローダ設定を使用しません。

## 18.2 必要なパッケージ

openSUSE® 上で kexec を再起動の高速化や潜在的なハードウェア問題を避けるために使用したい場合は、kexec-tools パッケージをインストールする必要があります。このパッケージには kexec-bootloader というスクリプトが含まれていて、ここからブートローダの設定を読み込み、kexec を通常のブートローダが行なうのと同じオプション設定でカーネルを起動することができます。kexec-bootloader -h を実行すると、利用可能なオプション一覧を表示することができます。

カーネルクラッシュの際に、有益なデバッグ情報を取得できるように環境を設定するには、さらに makedumpfile パッケージをインストールする必要があります。

openSUSE で kdump を使用する際には、YaST の Kdump モジュールを使用するのがお勧めです。シェルを起動して root の状態になり、zypper install yast2-kdump を実行し、yast2-kdump パッケージをインストールしてください。

## 18.3 kexec の内側

kexec で最も重要なコンポーネントは /sbin/kexec コマンドです。kexec を利用してカーネルを読み込む際、2 種類の方法で読み込むことができます：

- `kexec -l` カーネルイメージとして実行すると、通常の再起動のように、本番のカーネルが存在するアドレス領域にカーネルを読み込むことができます。読み込んだ後は、`kexec -e` でカーネルを起動することができます。

- `kexec -p` カーネルイメージ として実行すると、カーネルを予約領域内に読み込むことができます。これにより、このカーネルはシステムクラッシュ時に自動起動されるようになります。

システムがクラッシュした場合に他のカーネルを起動し、本番用としてそれまで使用していたカーネルのデータを保持したい場合は、システムメモリ内に専用の領域を予約しておく必要があります。このとき、本番用のカーネルはその領域を使用することはなく、常に空けたままの状態にしておきます。この領域はキャプチャ作業用のカーネルのもので、それまで使用していた本番カーネルの領域を保持するために用意します。メモリ領域の予約を行なうには、本番カーネルの起動時に `crashkernel = サイズ@オフセット` というパラメータを設定します。なお、このパラメータはキャプチャ作業用のカーネルパラメータではありません。キャプチャカーネル側では `kexec` を使用することはありません。

キャプチャカーネルは予約用の領域に読み込まれ、カーネルがクラッシュするのを待機します。`kdump` は本番カーネルがクラッシュすると、キャプチャカーネル側を実行しようとします。これは本番用に使用していたカーネルは、その時点では既に信頼できるものではないためです。このことから、場合によっては `kdump` が失敗する場合があります。

キャプチャカーネルを読み込むには、カーネルの起動パラメータ内にそれらを設定します。通常は初期 RAM ファイルシステム (`initrd`) を使用して設定します。初期 RAM ファイルシステムは `--initrd = ファイル名` の形式で指定します。また、`--append = コマンドライン` のように指定すると、起動するカーネルに対してコマンドラインオプションを追加することができます。これは本番用のカーネルに設定されていた、カーネルの起動に必要なオプションを設定するのに便利な機能です。本番用のカーネルに設定されているものをコピーするだけであれば `--append = "$(cat /proc/cmdline)"` と設定してもかまいませんし、`--append = "$(cat /proc/cmdline) 追加オプション"` として追加のものを設定してもかまいません。

なお、読み込み済みのカーネルはいつでも読み込みを解除することができます。`-l` オプションで読み込んだカーネルの読み込みを解除するには、`kexec -u` コマンドを使用します。`-p` オプションで読み込んだクラッシュカーネルの読み込みを解除するには、`kexec -p -u` コマンドを使用します。

## 18.4 基本的な kexec の使用方法

お使いの `kexec` 環境が正しく動作することを確認するには、下記の手順を実施します：

- 1 現在ログイン中のユーザが存在せず、重要なサービスについてもシステム上で動作していないことを確認します。
- 2 root でログインします。
- 3 telinit 1 を実行して、ランレベル 1 に移行します。
- 4 本番用のカーネル内にあるアドレス領域に対して、新しいカーネルを読み込むには、下記のように実行します:

```
kexec -l /boot/vmlinuz --append="$(cat /proc/cmdline)" --initrd=/boot/initrd
```

- 5 ルートファイルシステムを除き、すべてのマウント済みファイルシステムのマウントを `umount -a` で解除します。

---

### 重要: ルートファイルシステムのマウント解除について

ファイルシステムのマウントを解除する場合、よく発生するのが `device is busy` という警告メッセージです。ルートファイルシステムは、システムが動作中の場合にはマウントを解除できませんので、警告メッセージは無視してかまいません。

---

- 6 ルートファイルシステムを読み込み専用モードで再マウントします:

```
mount -o remount,ro /
```

- 7 `kexec -e` を実行し、ステップ 4 (202 ページ) で読み込んでおいたカーネルで起動を行ないます。

以前に読み書き可能な状態でマウントしていたディスクボリュームについて、これらのマウントを解除する作業が重要な点です。これは、`reboot` のシステムコールが呼び出されるとすぐに動作してしまうため、読み書き可能な状態でマウントされているハードディスクドライブのボリュームが存在すると、自動では同期もマウント解除も行なわれないためです。この場合は、新しいカーネルがそれらを「汚れた」状態にあるものとして認識します。読み込み専用のディスクボリュームと、仮想的なファイルシステムについては、マウントを解除する必要はありません。マウントを解除すべきファイルシステムの一覧を得たい場合は、`/etc/mtab` ファイルをお読みください。

新しい方のカーネルは、古いほうのカーネル内のアドレス領域内に読み込まれたあと、古いカーネルを置き換えて動作し、即時に制御を獲得します。これにより、通



常の起動メッセージが表示されます。また、新しいカーネルが起動すると、すべてのハードウェア初期化作業とファームウェアチェックが飛ばされます。このとき、一切の警告メッセージが表示されないことを確認してください。すべてのファイルシステムは、以前の段階でマウント解除されていれば、正常な (汚れてはいない) 状態になります。

## 18.5 kexec を頻繁な再起動用に設定する方法

kexec は頻繁に再起動するような環境でしばしば使用されます。また、たとえばハードウェアの検出ルーチンを実行するのに長い時間がかかる場合や、起動そのものが信頼できないような環境でも使用します。

---

### 注記: kexec を利用した再起動

openSUSE® の以前のバージョンでは、設定ファイル `/etc/sysconfig/shutdown` と `/etc/init.d/halt` を編集して、システムの再起動の際に kexec を利用するように設定する必要がありました。現行のバージョンではシステムファイルを手作業で編集する必要はありません。バージョン 11 以降であれば、既に kexec を利用するように設定されているためです。

---

なお、ブートローダとファームウェアについては、kexec を利用した再起動の際には使用されることはないことに注意してください。ブートローダに対して行なった変更が存在する場合、実際の (ハードウェアの) 再起動を行なわない限り、設定は無視されます。

## 18.6 基本的な kdump の設定

kdump を使用することで、カーネルのダンプを保存することができます。カーネルがクラッシュした場合、クラッシュした環境内のメモリイメージをファイルシステムにコピーしておくことで、カーネルクラッシュの原因を調べることができるためです。これを「コアダンプ」と呼びます。

kdump は kexec (詳しくは 第18章 *kexec* と *kdump* (199 ページ) をお読みください) に似た動作をします。キャプチャカーネルは本番用のカーネルがクラッシュした後に動作し始めるもので、kexec では本番用のカーネルがキャプチャカーネ

ルに置き換わる動作をしていましたが、kdump ではクラッシュした本番用のカーネルが存在するメモリ領域について、以前と同じくアクセスできるという点が異なります。このことから、kdump カーネルの環境内からクラッシュしたカーネルのメモリスナップショットを採取することができるようになっています。

kdump は手作業で設定することができるほか、YaST を利用して行なうこともできます。

## 18.6.1 手作業での kdump 設定

kdump は自身の設定を `/etc/sysconfig/kdump` ファイル から読み込みます。お使いのシステムで kdump が動作することを確認するには、まず既定の設定ファイルを利用して行なってください。kdump を既定の設定のまま使用するには、下記のような手順で行ないます：

- 1 お使いのブートローダの設定ファイルに、下記のようなカーネルのコマンドラインオプションを追加して、システムを再起動します：

`crashkernel=サイズ@オフセット`

それぞれ `サイズ` と `オフセット` の値には、下記の表に書かれている値を指定します：

**表 18.1** 追加のカーネルコマンドラインパラメータに対する推奨値

| アーキテクチャ         | 推奨値                                                                                          |
|-----------------|----------------------------------------------------------------------------------------------|
| i386 および x86-64 | <code>crashkernel=64M@16M</code>                                                             |
| IA64            | <code>crashkernel=256M</code> (小規模システムの場合) または<br><code>crashkernel=512M</code> (大規模システムの場合) |
| ppc64           | <code>crashkernel=128M</code> または<br><code>crashkernel=256M</code> (大規模システムの場合)              |

- 2 kdump 初期化スクリプトを有効にします：

```
chkconfig boot.kdump on
```

- 3 `/etc/sysconfig/kdump` 内のオプション設定を修正することもできます。それぞれ値の意味については、それぞれコメント欄に書かれている説明をお読みください。
- 4 `rckdump start` の初期化 スクリプトを実行するか、もしくはシステムを再起動します。

`kdump` を既定値で設定したあとは、期待通りに動作することを確認します。なお動作確認を行なう際には、お使いのシステムに誰もログインしていないこと、および重要なサービスが動作していないことを確認しておいてください。動作確認は下記のようにして行ないます:

- 1 `telinit 1` を実行して、ランレベル 1 に切り替えます。
- 2 ルートファイルシステムを除く、すべてのディスクのファイルシステムについてマウントを解除します。これは `umount -a` を実行すると実施することができます。
- 3 ルートファイルシステムについては、読み込み専用モードで再マウントします:  

```
mount -o remount,ro /
```
- 4 Magic SysRq キーに対する `procfs` のインターフェイスを利用して、「カーネルパニック」を発生させます:

```
echo c >/proc/sysrq-trigger
```

---

## 重要: カーネルダンプのサイズ

`KDUMP_KEEP_OLD_DUMPS` オプションでは、カーネルダンプを保持する個数を制御することができます (既定値は 5 です)。圧縮を行なわない場合、ダンプのサイズは最大で物理メモリの搭載量に等しくなります。`/var` のパーティション内に必要な容量があるかどうかを確認してください。

---

キャプチャカーネルが起動すると、クラッシュしたほうのカーネルについて、メモリのスナップショットがファイルシステム内に保存されます。保存先のパスは `KDUMP_SAVEDIR` オプションで設定し、既定では `/var/crash` に設定されています。また、`KDUMP_IMMEDIATE_REBOOT` を `yes` に設定すると、システムはその後本番用のカーネルを利用して再起動します。再起動が完了したらログインを行なうと、`/var/crash` 内のダンプを確認することができるようになります。

---

## 警告: X11 セッション内での画面フリーズ

X11 セッションにログインしている状態で kdump の制御下に移行すると、画面には全く表示がなされないまま画面の動きが止まり (フリーズし) ます。kdump の動きのうちのいくつかは表示から確認することができます (たとえば画面上に起動中のカーネルに関する形の崩れたメッセージが現われたりします)。

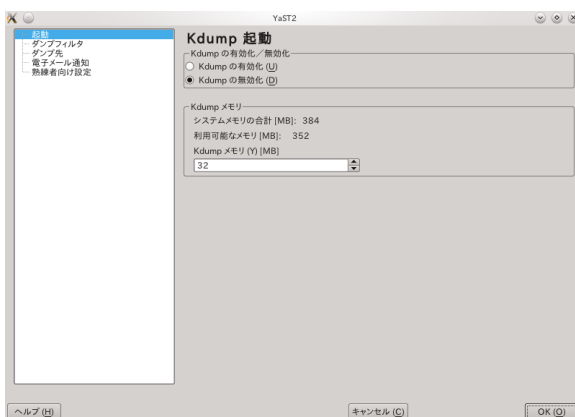
このような場合でも、コンピュータをリセットしたりはしないでください。kdump が動作を完了するのにしばらくの時間が必要であるためです。

---

## 18.6.2 YaST の設定

kdump を YaST から設定するには、まず `yast2-kdump` パッケージをインストールする必要があります。インストールが完了したら、`root` から YaST コントロールセンター の システム カテゴリ内にある カーネル *Kdump* モジュールを起動するか、もしくは コマンドラインで `yast2 kdump` と入力します。

**図 18.1** YaST2 kdump モジュール - スタートアップページ



まず *起動* ウィンドウでは、*Kdump* の有効化 を選択します。kdump メモリについては既定値のままでかまいません。ほとんどの システムで十分な値になっているためです。

次に左側の一覧で *ダンプフィルタ* を選択します。ここでは *ダンプ* にどのページを含めるかを設定します。カーネルの問題をデバッグするに あたっては、下記のメモリ内容については含めなくてもかまいません:

- ゼロで埋められたページ
- ページのキャッシュ
- ユーザデータページ
- 使われていないページ

さらに **ダンプ先** のウィンドウでは、ダンプの保存先の種類と保存先の URL を指定します。FTP や SSH などのネットワークプロトコルを選択した場合は、必要なアクセス情報についても設定を行なう必要があります。

続いて **電子メール通知** のウィンドウでは、kdump に対して 電子メールを介した通知を送信させるための設定を行なうことができます。各項目を設定し終えたら、OK を押してウィンドウを閉じてください。なお、**熟練者向け設定** ウィンドウを利用してより細かい設定を行なうこともできます。これで kdump の設定は完了です。

## 18.7 クラッシュダンプの解析

ダンプファイルを採取したら、次はそれを解析する段階です。これにはいくつかの方法があります。

ダンプを解析する際のもっとも原始的なツールが GDB です。もちろん最新の環境でも GDB を利用することができますが、いくつかの不都合や制限があります：

- GDB はカーネルダンプのデバッグ用に特化した作りにはなっていません。
- GDB は 32 ビットプラットフォームにおいて、ELF64 バイナリには対応していません。
- GDB は ELF ダンプ以外の形式を解釈できません (圧縮ダンプにも対応していません)。

上記のような問題により、*crash* ユーティリティが作成されるようになりました。*crash* ユーティリティはクラッシュダンプを解析するほか、動作中のシステムからでもデバッグを行なうことができます。また、Linux カーネルのデバッグ固有の機能も提供されていて、より高度なデバッグ作業に便利な作りになっています。

Linux カーネルをデバッグしたい場合は、デバッグ情報のパッケージについてもあらかじめインストールしておく必要があります。お使いのシステムに該当するパッ

ケージがインストールされているかどうかを確認するには、`zypper se kernel | grep debug` を実行してください。

---

### 重要: デバッグ情報のパッケージリポジトリ

お使いのシステムでオンライン更新の購読を行なっている場合、「debuginfo」パッケージは `*-Debuginfo-Updates` という名称のオンラインインストールリポジトリ内に存在しています。YaST を利用してリポジトリを有効に設定してください。

---

お使いのマシンで採取されたダンプを `crash` で開くには、下記のようなコマンドを実行します:

```
crash /boot/vmlinux-2.6.32.8-0.1-default.gz /var/
crash/2010-04-23-11¥:17/vmcore
```

最初のパラメータではカーネルイメージを指定します。2 つめのパラメータでは `kdump` で採取されたダンプファイルを指定します。ダンプファイルは、既定では `/var/crash` ディレクトリ内に保存されます。

## 18.7.1 カーネルのバイナリフォーマット

Linux カーネルは Executable and Linkable Format (ELF) という形式で作成されています。このファイルは通常 `vmlinux` と呼ばれ、コンパイル処理の際に直接作成されます。すべてのブートローダでサポートされているというわけではありません (特に x86 環境 (i386, x86\_64) 以外) では ELF バイナリに対応しています。openSUSE® でサポートされる アーキテクチャでは、それぞれ下記のような解決方法があります。

### 18.7.1.1 x86 (i386 および x86\_64)

多くは歴史上の経緯によるものですが、Linux カーネルは 2 種類のパートから構成されます: 1 つは Linux カーネルそれ自身 (`vmlinux`)、もう 1 つはブートローダが実行するセットアップコードです。

これら 2 つのパートは `bzImage` と呼ばれるファイルにまとめられます。このファイルはカーネルのソースツリー内に存在するはずのものです。このファイルは現在では、`vmlinuz` (x ではなく z であることに注意してください) とカーネルパッケージ内で呼ばれます。

ELF イメージは x86 環境では直接使用されることはありません。そのため、カーネルパッケージの本体である `vmlinux` ファイルは、圧縮された `vmlinux.gz` という形式で配置されます。

要約すると、x86 環境の SUSE カーネルパッケージには、2 種類の カーネルファイルが存在することになります：

- ブートローダで実行される `mlinuz` ファイル。
- `crash` や GDB で利用する圧縮 ELF イメージ `vmlinux.gz` ファイル。

## 18.7.1.2 IA64

IA64 アーキテクチャ上で Linux カーネルを起動するためのブートローダ `elilo` では、特に何らかの作業を行なわなくても ELF 形式の読み込みに対応しています (圧縮されているものにも対応しています)。IA64 のカーネルパッケージには `mlinuz` と呼ばれる 圧縮 ELF イメージのファイルのみが含まれていて、これは x86 環境における `vmlinux.gz` ファイルと同じ形式のものになっています。

## 18.7.1.3 PPC および PPC64

PPC 環境では `yaboot` ブートローダを使用しますが、このブートローダは ELF イメージの読み込みには対応しているものの、圧縮された 形式には対応していません。そのため、PPC 版のカーネルパッケージには、ELF 形式の Linux カーネルファイル `vmlinux` だけが 存在しています。`crash` を使用するにあたっては、もっとも扱いやすいアーキテクチャであると言えます。

他のマシンでダンプを解析しようとしている場合は、まずコンピュータの アーキテクチャと、デバッグに必要なファイルが存在しているかどうかを確認しなければなりません。

他のコンピュータでダンプを解析するにあたっては、そのマシンでは同じ アーキテクチャの Linux システムを動作させる必要があります。互換性を確認 するには、両方のコンピュータで `uname -i` を実行し、出力が同じであることを確認してください。

なお、他のコンピュータでダンプを解析する場合は、`kernel` や `kernel debug` パッケージに含まれるファイルも 必要となることに注意してください。

- 1 まずはカーネルダンプと、`/boot` ディレクトリにあるカーネルイメージを用意します。また、関連するデバッグ情報ファイルを `/usr/lib/debug/boot` ディレクトリから専用の 単一ディレクトリにコピーします。

- 2 あわせて `/lib/modules/$(uname -r)/kernel/` ディレクトリにあるカーネルモジュールと、`/usr/lib/debug/lib/modules/$(uname -r)/kernel/` ディレクトリにある関連するデバッグ情報ファイルをコピーします。コピー先は `modules` というサブディレクトリに行なってください。
- 3 ダンプとカーネルイメージ、およびカーネルのデバッグ情報ファイル、そして `modules` サブディレクトリが存在する状態で、`crash` ユーティリティを起動します：  
`crash vmlinux-(バージョン) vmcore`

---

### 注記: カーネルイメージに対するサポート

圧縮されたカーネルイメージ (gzip による圧縮形式で、`bzImage` ファイルは 除きます) への対応については、SUSE パッケージでは openSUSE® 11 以降のバージョンで対応しています。古いバージョンをお使いの場合は、`vmlinux.gz` (x86 アーキテクチャの場合) または `vmlinuz` (IA64 アーキテクチャの場合) の圧縮を解除し、`vmlinux` に展開しておく必要があります。

---

どのコンピュータでダンプの解析を行なう場合であっても、`crash` ユーティリティは下記のような出力を行ないます:

```
tux@mercury:~> crash /boot/vmlinux-2.6.32.8-0.1-default.gz
/var/crash/2010-04-23-11¥:17/vmcore
```

```
crash 4.0-7.6
Copyright (C) 2002, 2003, 2004, 2005, 2006, 2007, 2008 Red Hat, Inc.
Copyright (C) 2004, 2005, 2006 IBM Corporation
Copyright (C) 1999-2006 Hewlett-Packard Co
Copyright (C) 2005, 2006 Fujitsu Limited
Copyright (C) 2006, 2007 VA Linux Systems Japan K.K.
Copyright (C) 2005 NEC Corporation
Copyright (C) 1999, 2002, 2007 Silicon Graphics, Inc.
Copyright (C) 1999, 2000, 2001, 2002 Mission Critical Linux, Inc.
This program is free software, covered by the GNU General Public License,
and you are welcome to change it and/or distribute copies of it under
certain conditions. Enter "help copying" to see the conditions.
This program has absolutely no warranty. Enter "help warranty" for details.
```

```
GNU gdb 6.1
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "x86_64-unknown-linux-gnu"...
```

```
KERNEL: /boot/vmlinux-2.6.32.8-0.1-default.gz
DEBUGINFO: /usr/lib/debug/boot/vmlinux-2.6.32.8-0.1-default.debug
DUMPFILE: /var/crash/2009-04-23-11:17/vmcore
```



```

 CPUS: 2
 DATE: Thu Apr 23 13:17:01 2010
 UPTIME: 00:10:41
LOAD AVERAGE: 0.01, 0.09, 0.09
 TASKS: 42
 NODENAME: eros
 RELEASE: 2.6.32.8-0.1-default
 VERSION: #1 SMP 2010-03-31 14:50:44 +0200
 MACHINE: x86_64 (2999 Mhz)
 MEMORY: 1 GB
 PANIC: "SysRq : Trigger a crashdump"
 PID: 9446
 COMMAND: "bash"
 TASK: ffff88003a57c3c0 [THREAD_INFO: ffff880037168000]
 CPU: 1
 STATE: TASK_RUNNING (SYSRQ)
crash>

```

コマンドの出力では、最初に有益な情報が出力されます。上記の例では、クラッシュ時点で 42 個のタスクが存在していて、クラッシュの原因はプロセス ID (PID) 9446 による SysRq トリガーであることが示されています。また、原因となったプロセスは Bash プロセスであることが示されています。これは前述の例で、echo コマンドという Bash シェルの内部コマンドを利用したことによるものです。

*crash* ユーティリティはその後 GDB を起動し、多数の追加 コマンドを提供します。たとえば bt コマンドをパラメータ無しで入力すると、クラッシュが発生した時点でのタスクに対して、バックトレースを表示させることができます:

```

crash> bt
PID: 9446 TASK: ffff88003a57c3c0 CPU: 1 COMMAND: "bash"
#0 [ffff880037169db0] crash_kexec at ffffffff80268fd6
#1 [ffff880037169e80] __handle_sysrq at ffffffff803d50ed
#2 [ffff880037169ec0] write_sysrq_trigger at ffffffff802f6fc5
#3 [ffff880037169ed0] proc_reg_write at ffffffff802f068b
#4 [ffff880037169f10] vfs_write at ffffffff802b1aba
#5 [ffff880037169f40] sys_write at ffffffff802b1c1f
#6 [ffff880037169f80] system_call_fastpath at ffffffff8020bfb6
RIP: 00007fa958991f60 RSP: 00007fff61330390 RFLAGS: 00010246
RAX: 0000000000000001 RBX: ffffffff8020bfb6 RCX: 0000000000000001
RDX: 0000000000000002 RSI: 00007fa959284000 RDI: 0000000000000001
RBP: 0000000000000002 R8: 00007fa9592516f0 R9: 00007fa958c209c0
R10: 00007fa958c209c0 R11: 0000000000000246 R12: 00007fa958c1f780
R13: 00007fa959284000 R14: 0000000000000002 R15: 00000000595569d0
ORIG_RAX: 0000000000000001 CS: 0033 SS: 002b
crash>

```

ここでようやく何が行なっているのが明確になります。内部コマンドである echo は /proc/sysrq-trigger というファイルに対して文字を送信します。対応するハンドラがその文字を認識し、crash\_kexec() 関数を呼び出します。この関数は

panic() を呼び出し、kdump がダンプを保存する動作を行っていたことになります。

基本的な GDB コマンドや bt の拡張版に加え、crash ユーティリティでは Linux カーネルの構造に対応した数多くのコマンドが用意されています。これらのコマンドは Linux カーネルの内部データ構造を理解するもので、それらの内容を人間にとって読み取りやすい形式で表示します。たとえばクラッシュが発生した時点でのタスク一覧を表示させたい場合は、ps コマンドを利用します。また sym コマンドでは、すべてのカーネルシンボル情報と対応するアドレスを一覧表示することができますほか、個別のシンボルに対してアドレスを表示することもできます。それ以外にも files コマンドでは、プロセスが開いているすべてのファイルディスクリプタを表示することができます。さらに kmem では、カーネルのメモリ使用に関する詳細な情報を得ることもできます。それ以外にも、vm コマンドではプロセスの仮想メモリや個別のページマッピングに対して調査を行なうことができます。これ以外にも便利なコマンドが多数用意されていて、様々なオプション指定を行なうことができるようになっています。

上述のコマンドは、いずれも一般的な Linux コマンドである ps や lsof などの機能に対応しています。デバグガを利用して正確なイベント順序を調査したい場合は、GDB の使い方に関する知識を得て、高度なデバグ 技術を習得しておく必要があります。これらの知識や技術は本ドキュメントの範疇外であるため、説明は行なっていません。加えて、Linux カーネルに関する知識も必要となります。いくつかの有益な参照情報源については、このドキュメントの末尾に示しています。

## 18.8 高度な kdump 設定

kdump の設定は /etc/sysconfig/kdump ファイル内に保存されています。YaST を利用して設定することもできます。YaST を利用した設定を行なう場合は、YaST コントロールセンター から システム > カーネル *Kdump* を選択してください。それぞれ下記に示す kdump オプションが便利でしょう:

カーネルダンプを出力するディレクトリを指定するには、KDUMP\_SAVEDIR オプションを使用します。なお、カーネルダンプのファイルサイズは非常に大きくなる可能性があることに注意してください。kdump は、ディスクの空き容量からダンプファイルの見積もりサイズを引いた値が KDUMP\_FREE\_DISK\_SIZE オプションで指定した値よりも少ない場合、ダンプの保存を拒否します。また、KDUMP\_SAVEDIR では URL 形式 (プロトコル://仕様) で指定を行なうことができます。ここで プロトコル は file, ftp, sftp, nfs, cifs のいずれかを指定し、仕様にはそれぞれのプロトコルにあった指定を行ないます。たとえば FTP サーバ上にカーネルダンプを保存する

には、ftp://ユーザ名:パスワード@ftp.example.com:123/var/crash のように指定します。

カーネルダンプは前述の通り非常に大きくなるものであり、分析には不要な多くのページ情報が含まれています。KDUMP\_DUMPLEVEL オプションでは、不要なページ情報を省略するかどうかを設定することができます。このオプションでは 0 から 31 までの整数を指定します。0 を指定した場合は、ダンプファイルが最も大きくなります。逆に 31 を指定すると、最も小さいダンプファイルを生成する意味になります。指定可能な値に関する詳細情報については、kdump のマニュアルページ (man 7 kdump) をお読みください。

また、カーネルダンプのサイズを小さくする方法がもう 1 つあります。たとえばネットワークを介してダンプを転送する場合や、ダンプディレクトリにディスク領域を残しておきたいような場合は、KDUMP\_DUMPFORMAT に *compressed* (圧縮する) を設定してください。crash ユーティリティでは、圧縮されたダンプに対する動的な展開にも対応しています。

---

### 重要: kdump 設定ファイルの変更

/etc/sysconfig/kdump ファイルを手作業で編集した場合は、編集後に rckdump restart を実行する必要があります。これを実行しない場合、システムを再起動するまで変更点が反映されません。

---

## 18.9 さらなる情報

本ドキュメントでは kexec や kdump の使用方法に関してすべての範囲を網羅できていません。それぞれ下記に示す情報資源を参照して必要な情報を得てください:

- kexec ユーティリティの使用方法については、kexec のマニュアル ページ (man 8 kexec) をお読みください。
- kexec に関する一般的な情報については、<http://www.ibm.com/developerworks/jp/linux/library/l-kexec/index.html> をお読みください。リンク先の情報は少し古いものになっています。
- kdump のうち、SUSE Linux 固有の箇所に関する情報は、<http://ftp.suse.com/pub/people/tiwai/kdump-training/kdump-training.pdf> (英語) をお読みください。

- kdump の内部に関するより深い情報については、<http://lse.sourceforge.net/kdump/documentation/ols2oo5-kdump-paper.pdf> (英語) をお読みください。

*crash* によるダンプ分析や、デバッグツールに関する詳細な情報は、それぞれ下記をお読みください:

- GDB の info ページ (info gdb) に加え、印刷もできる形式の ガイド資料が <http://sourceware.org/gdb/documentation/> (英語) で公開されています。
- crash ユーティリティの使用方法に関する説明用のホワイトペーパーが [http://people.redhat.com/anderson/crash\\_whitepaper/](http://people.redhat.com/anderson/crash_whitepaper/) (英語) で公開されています。
- crash ユーティリティでは完全なオンラインヘルプも用意されています。特定のコマンド に対してオンラインヘルプを表示させたい場合は、help コマンド を実行してください。
- 必要な Perl 技術をお持ちの場合は、Alicia を利用してデバッグを簡素化することも できます。この Perl ベースの crash ユーティリティ向けフロントエンドは、<http://alicia.sourceforge.net/> (英語) で公開されています。
- Perl ではなく Python をご希望の場合は、Pykdump というツールをインストールすることも できます。このパッケージは Python スクリプトから GDB を制御することができるもので、<http://sf.net/projects/pykdump> (英語) から ダウンロード可能です。
- Linux カーネルの内部に関する広範囲の概要を知りたい場合は、Daniel P. Bovet 氏と Marco Cesati 氏が書かれた *詳解 Linuxカーネル* (原文: ISBN 978-0-596-00565-8) (日本語訳: ISBN 978-4873113135) を お読みください。



# GNU ライセンス

本付録には、GNU General Public License バージョン 2 と GNU Free Documentation License バージョン 1.2 を掲載しています。

なお、八田真行氏 (mhatta@gnu.org) [<mailto:mhatta@gnu.org>] による各ライセンスの日本語訳を併記しています。

ただし、各日本語訳は *非公式*なものであり、フリーソフトウェア財団 (the Free Software Foundation) によって発表されたものではないことにご注意ください。法的に有効なものは常に原文 (つまり英語版) 側であり、日本語訳は各ライセンスをよりよく理解する支援を行なう目的で 作成されたもの、という扱いです。

また、日本語訳は DocBook (novdoc) に合わせて段落を分割しているほか、引用符のタグ化 ("blah" -> <quote>blah</quote>) とリンクの生成 (ulink) を行なっています。

## GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

## GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

**0.** This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The «Program», below, refers to any such program or work, and a «work based on the Program» means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term «modification».) Each licensee is addressed as «you».

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

**1.** You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

**2.** You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a)** You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- b)** You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- c)** If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

**3.** You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- a)** Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

- b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and [any later version], you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

*NO WARRANTY*

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

## *END OF TERMS AND CONDITIONS*

### How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and an idea of what it does.
Copyright (C) yyyy name of author
```

```
This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type `show w' . This is free software, and you are welcome
to redistribute it under certain conditions; type `show c'
for details.
```

The hypothetical commands "show w" and "show c" should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than "show w" and "show c"; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
```



interest in the program `Gnomovision'  
(which makes passes at compilers) written  
by James Hacker.

signature of Ty Coon, 1 April 1989  
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License [<http://www.fsf.org/licenses/lgpl.html>] instead of this License.

## GNU 一般公衆利用許諾契約書 (日本語訳)

バージョン 2, 1991年6月

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

この利用許諾契約書を、一字一句そのままに複製し頒布することは許可する。しかし変更は認めない。

### はじめに

ソフトウェア向けライセンスの大半は、あなたがそのソフトウェアを共有したり 変更したりする自由を奪うように設計されています。対照的に、GNU 一般公衆利用許諾契約書は、あなたがフリーソフトウェアを共有したり変更したりする自由を保証する--すなわち、ソフトウェアがそのユーザすべてにとってフリー であることを保証することを目的としています。この一般公衆利用許諾契約書は、フリーソフトウェア財団のソフトウェアのほとんどに適用されており、また GNU GPLを適用すると決めたフリーソフトウェア財団以外の作者によるプログラムにも適用されています(いくつかのフリーソフトウェア財団のソフトウェアには、GNU GPLではなくGNU ライブラリー一般公衆利用許諾契約書が適用されています)。あなたもまた、ご自分のプログラムにGNU GPLを適用することが可能です。

私たちがフリーソフトウェアと言うとき、それは利用の自由について言及している のであって、価格は問題にしていません。私たちの一般公衆利用許諾契約書は、あなたがフリーソフトウェアの複製物を頒布する自由を保証するよう設計されています (希望に応じてその種のサービスに手数料を課す自由も保証されます)。また、あなたがソースコードを受け取るか、あるいは望めばそれを入手することが可能であるということ、あなたがソフトウェアを変更し、その一部を新たなフリーの プログラムで利用できるとのこと、そして、以上で述べたようなことができる ということがあなたに知られるということも保証されます。

あなたの権利を守るため、私たちは誰かがあなたの有するこれらの権利を否定する ことや、これらの権利を放棄するよう要求することを禁止するという制限を加える 必要があります。よって、あなたがソフトウェアの複製物を頒布したりそれを変更 したりする場合には、そういった制限のためにあなたにある種の責任が発生することになります。

例えば、あなたがフリーなプログラムの複製物を頒布する場合、有料が無料に 関わらず、あなたは自分が有する権利を全て受領者に与えなければなりません。また、あなたは彼らもソースコードを受け取るか手に入れることができるよう 保証しなければなりません。そして、あなたは彼らに対して以下で述べる条件を示し、彼らに自らの持つ権利について知らしめるようにしなければなりません。

私たちはあなたの権利を二段階の手順を踏んで保護します。(1) まずソフトウェアに対して著作権を主張し、そして (2) あなたに対して、ソフトウェアの複製や頒布または改変についての法的な許可を 与えるこの契約書を提示します。

また、各作者や私たちを保護するため、私たちはこのフリーソフトウェアには 何の保証も無いということを誰もが確実に理解するようにし、またソフトウェアが 誰か他人によって改変され、それが次々と頒布されていったとしても、その受領者は 彼らが手に入れたソフトウェアがオリジナルのバージョンでは無いこと、そして 原作者の名声は他人によって持ち込まれた可能性のある問題によって影響される ことがないということを周知させたいと思います。

最後に、ソフトウェア特許がいかなるフリーのプログラムの存在にも不断の脅威を 投げかけていますが、私たちは、フリーなプログラムの再頒布者が個々に特許 ライセンスを取得することによって、事実上プログラムを独占的にしてしまうという 危険を避けたいと思います。こういった事態を予防するため、私たちはいかなる特許も 誰もが自由に利用できるようライセンスされるか、全くライセンスされないかの どちらかでなければならないことを明確にしました。

(訳注: 本契約書で「独占的(proprietary)」とは、ソフトウェアの利用や再頒布、改変が禁止されているか、許可を得ることが必要とされているか、あるいは厳しい 制限が課せられていて自由にそうすることが事実上できなくなっている状態のことを 指す。詳しくは <http://www.gnu.org/philosophy/categories.ja.html#ProprietarySoftware> [<http://www.gnu.org/philosophy/categories.ja.html#ProprietarySoftware>] を参照せよ。)

複製や頒布、改変についての正確な条件と制約を以下で述べていきます。

### 複製、頒布、改変に関する条件と制約

**0.** この利用許諾契約書は、そのプログラム(またはその他の著作物)を この一般公衆利用許諾契約書の定める条件の下で頒布できる、という告知が 著作権者によって記載されたプログラムまたはその他の著作物全般に適用される。以下では、「プログラム」とはそのようにしてこの契約書が適用され プログラムや著作物全般を意味し、また「プログラムを基にした著作物」とは「プログラム」やその他の著作権法の下で派生物と見なされるもの 全般を指す。すなわち、「プログラム」かその一部を、全く同一の ままか、改変を加えたか、あるいは他の言語に翻訳された形で含む

著作物のことである(「改変」という語の本来の意味からはずれるが、以下では 翻訳も改変の一種と見なす)。それぞれの契約者は「あなた」と表現される。

複製や頒布、改変以外の活動はこの契約書ではカバーされない。それらはこの契約書の 対象外である。「プログラム」を実行する行為自体に制限はない。また、そのような「プログラム」の出力結果は、その内容が「プログラム」を基にした著作物を構成する場合のみ この契約書によって保護される(「プログラム」を実行したことによって 作成されたということは無関係である)。このような線引きの妥当性は、「プログラム」が何を  
するのかに依存する。

1. それぞれの複製物において適切な著作権表示と保証の否認声明(disclaimer of warranty) を目立つよう適切に掲載し、またこの契約書および一切の保証の不在に触れた 告知すべてをそのまま残し、そしてこの契約書の複製物を「プログラム」のいかなる受領者にも「プログラム」と共に頒布する限り、あなたは「プログラム」のソースコードの複製物を、あなたが受け取った通りの形で複製または頒布することができる。媒体は問わない。

あなたは、物理的に複製物を譲渡するという行為に関して手数料を課しても良いし、希望によっては手数料を取って交換における保護の保証を提供しても良い。

2. あなたは自分の「プログラム」の複製物かその一部を改変して「プログラム」を基にした著作物を形成し、そのような改変点や 著作物を上記第1節の定める条件の下で複製または頒布することができる。ただし、そのためには以下の条件すべてを満たしていなければならない:

a) あなたがそれらのファイルを変更したということと変更した日時が良く 分かるよう、改変されたファイルに告示しなければならない。

b) 「プログラム」またはその一部を含む著作物、あるいは「プログラム」かその一部から派生した著作物を頒布あるいは 発表する場合には、その全体をこの契約書の条件に従って第三者へ無償で 利用許諾しなければならない。

c) 改変されたプログラムが、通常実行する際に対話的にコマンドを読むようになっているならば、そのプログラムを最も一般的な方法で対話的に実行する際、適切な著作権表示、無保証であること(あるいはあなたが保証を提供するということ)、ユーザがプログラムをこの契約書で述べた条件の下で頒布することができる、ということ、そしてこの契約書の複製物を閲覧するにはどうしたらよいかというユーザへの説明を含む告知が印刷されるか、あるいは画面に表示される ようにしなければならない(例外として、「プログラム」そのものは対話的であっても通常そのような告知を印刷しない場合には、「プログラム」を基にしたあなたの著作物に そのような告知を 印刷させる必要はない)。

以上の必要条件是全体としての改変された著作物に適用される。著作物の一部が「プログラム」から派生したのではないと確認でき、それら自身 別の独立した著作物であると合理的に考えられるならば、あなたがそれらを別の 著作物として分けて頒布する場合、そういった部分にはこの契約書とその条件は 適用されない。しかし、あなたが同じ部分を「プログラム」を基にした 著作物全体の一部として頒布するならば、全体としての頒布物は、この契約書が 課す条件に従わなければならない。というのは、この契約書が他の契約者に与える 許可は「プログラム」丸ごと全体に及び、誰が書いたかは関係なく各部分のすべてを保護するからである。

よって、すべてあなたによって書かれた著作物に対し、権利を主張したりあなたの 権利に異議を申し立てることはこの節の意図するところではない。むしろ、その趣旨は「プログラム」を基にした派生物ないし集合著作物の 頒布を管理する権利を行使することにある。

また、「プログラム」を基にしていないその他の著作物を「プログラム」(あるいは「プログラム」を基にした著作物)と一緒に集めただけのものを一巻の保管装置ないし頒布媒体に収めても、その他の 著作物までこの契約書が保護する対象になるということにはならない。

3. あなたは上記第1節および2節の条件に従い、「プログラム」(あるいは 第2節における派生物)をオブジェクトコードないし実行形式で複製または頒布 することができる。ただし、その場合あなたは以下のうちどれか一つを実施 しなければならない:

a) 著作物に、『プログラム』に対応した完全かつ機械で読み取り可能な ソースコードを添付する。ただし、ソースコードは上記第1節および2節の条件に従いソフトウェアの交換で習慣的に使われる媒体で頒布しなければならない。あるいは、

b) 著作物に、いかなる第三者に対しても、『プログラム』に対応した完全かつ 機械で読み取り可能なソースコードを、頒布に要する物理的コストを上回らない 程度の手数料と引き換えに提供する旨述べた少なくとも3年間は有効な書面 になった申し出を添える。ただし、ソースコードは上記第1節および2節の条件に 従いソフトウェアの交換で習慣的に使われる媒体で頒布しなければならない。あるいは、

c) 対応するソースコード頒布の申し出に際して、あなたが得た情報を一緒に引き渡す (この選択肢は、営利を目的としない頒布であって、かつあなたが上記小節bで 指定されているような申し出と共にオブジェクトコードあるいは実行形式の プログラムしか入手していない場合に限り許可される)。

著作物のソースコードとは、それに対して改変を加える上で好ましいとされる 著作物の形式を意味する。ある実行形式の著作物にとって完全なソースコードとは、それが含むモジュールすべてのソースコード全部に加え、関連するインターフェース 定義ファイルのすべてとライブラリのコンパイルやインストールを制御するために 使われるスクリプトをも加えたものを意味する。しかし特別な例外として、そのコンポーネント自体が実行形式に付随するのでは無い限り、頒布されるものの 中に、実行形式が実行されるオペレーティングシステムの主要なコンポーネント (コンパイラやカーネル等)と通常一緒に(ソースかバイナリ形式のどちらかで) 頒布されるものを含んでいる必要はないとする。

実行形式またはオブジェクトコードの頒布が、指定された場所からコピーするための アクセス手段を提供することで為されるとして、その上でソースコードも同等の アクセス手段によって同じ場所からコピーできるようになっているならば、第三者が オブジェクトコードと一緒にソースも強制的にコピーせられるようになっていなくてもソースコード頒布の条件を満たしているものとする。

4. あなたは「プログラム」を、この契約書において明確に提示された 行為を除き複製や改変、サブライセンス、あるいは頒布してはならない。他に「プログラム」を複製や改変、サブライセンス、あるいは頒布する 企てはすべて無効であり、この契約書の下でのあなたの権利を自動的に終結させることになる。しかし、複製物や権利をこの契約書に従ってあなたから得た人々に 関しては、そのような人々がこの契約書に完全に従っている限り彼らのライセンスまで 終結することはない。

5. あなたはこの契約書を受諾する必要は無い。というのは、あなたはこれに署名していないからである。しかし、この契約書以外にあなたに対して「プログラム」やその派生物を改変または頒布する許可を与えるものは 存在しない。これらの行為は、あなたがこの契約書を受け入れない限り

法によって 禁じられている。そこで、「プログラム」(あるいは「プログラム」を基にした著作物全般)を改変しない頒布することにより、あなたは自分がそのような行為を行うためにこの契約書を受諾したということ、そして「プログラム」とそれに基づく著作物の複製や頒布、改変について この契約書が課す制約と条件をすべて受け入れたということを示したものと見なす。

6. あなたが「プログラム」(または「プログラム」を基にした著作物全般)を再頒布するたびに、その受領者は元々のライセンス許可者から、この契約書で指定された条件と制約の下で「プログラム」を複製や頒布、あるいは改変する許可を自動的に得るものとする。あなたは、受領者がここで認められた権利を行使することに関してこれ以上他のいかなる制限も課してはならない。あなたには、第三者がこの契約書に従うことを強制する責任はない。

7. 特許侵害あるいはその他の理由(特許関係に限らない)から、裁判所の判決あるいは 申し立ての結果としてあなたに(裁判所命令や契約などにより)このライセンスの 条件と矛盾する制約が課された場合でも、あなたがこの契約書の条件を免除される わけではない。もしこの契約書の下であなたに課せられた責任と他の関連する責任を 同時に満たすような形で頒布できないならば、結果としてあなたは「プログラム」を頒布することが全くてできないことである。例えば特許ライセンスが、あなたから直接間接を問わずコピーを受け取った人が 誰でも「プログラム」を使用 料無料で再頒布することを認めていない場合、あなたがその制約とこの契約書を両方とも満たすには「プログラム」の頒布を完全に中止するしかないだろう。

この節の一部分が特定の状況の下で無効ないし実施不可能な場合でも、節の残りの 部分は適用されるよう意図されている。その他の状況では 節が全体として適用されるよう 意図されている。

特許やその他の財産権を侵害したり、そのような権利の主張の効力に異議を唱えたり するようあなたを誘惑することがこの節の目的ではない。この節には、人々によって ライセンス慣行として実現されてきた、フリーソフトウェア頒布のシステムの完全性を 護るという目的しかない。多くの人々が、フリーソフトウェアの頒布システムが首尾一貫して適用されているという信頼に基づき、このシステムを通じて頒布される多様な ソフトウェアに 寛大な貢献をしてきたのは事実であるが、人がどのようなシステムを 通じてソフトウェアを頒布したいと思うかはあくまでも作者/寄与者次第であり、あなたが選択を押しつけることはできない。

この節は、この契約書のこの節以外の部分の一帰結になると考えられるケースを 徹底的に明らかにすることを目的としている。

8. 「プログラム」の頒布や利用が、ある国においては特許または著作権が 主張されたインターフェースのいずれかによって制限されている場合、「プログラム」にこの契約書を適用した元の著作権者は、そういった 国々を排除した明確な地理的頒布制限を加え、そこで排除されていない国の中や それらの国々の間でのみ頒布が許可されるようにしても構わない。その場合、そのような制限はこの契約書本文で書かれているのと同様に 見なされる。

9. フリーソフトウェア財団は、時によって改訂または新版の一般公衆利用許諾書を 発表することができる。そのような新版は現在のバージョンとその精神においては 似たものになるだろうが、新たな問題や懸念を解決するため細部では異なる可能性がある。

それぞれのバージョンには、見分けが付くようにバージョン番号が振られている。「プログラム」においてそれに適用されるこの契約書のバージョン番号が 指定されていて、更に「それ以降のいかなるバージョン(any later version)」も適用して 良いとなっていた場合、あなたは従う条件と制約として、指定のバージョンが、フリーソフトウェア財団によって発行された指定のバージョン以降の版のどれか一つの どちらかを選ぶことが出来る。「プログラム」でライセンスのバージョン番号が 指定されていないならば、あなたは今までにフリーソフトウェア財団から発行された バージョンの中から好きに選んで構わない。

10. もしあなたが「プログラム」の一部を、その頒布条件がこの契約書と 異なる他のフリーなプログラムと統合したいならば、作者に連絡して許可を求めよ。フリーソフトウェア財団が著作権を保有するソフトウェアについては、フリーソフトウェア財団に連絡せよ。私たちは、このような場合のために特別な例外を 設けることもある。私たちが決定を下すにあたっては、私たちのフリーソフトウェアの 派生物すべてがフリーな状態に保たれるということと、一般的にソフトウェアの共有と 再利用を促進するという二つの目標を規準に検討されるであろう。

## 無保証について

11. 「プログラム」は代価無しに利用が許可されるので、適切な法が 認める限りにおいて、「プログラム」に関するいかなる保証も 存在しない。書面で別に述べる場合を除いて、著作権者、またはその他の団体は、「プログラム」を、表明されたか言外にかかわらず、商業的適性を保証する ほどのめかしやある特定の目的への適合性(に限られない)を含む一切の 保証無しに「あるがまま」で提供する。「プログラム」の質と 性能に関する リスクのすべてはあなたに帰属する。「プログラム」に欠陥があると判明した場合、あなたは必要な保守点検や補修、修正に要する コストのすべてを引き受けることになる。

12. 適切な法が書面での同意によって命ぜられない限り、著作権者、または上記で 許可されている通りに「プログラム」を改変または再頒布した その他の団体は、あなたに対して「プログラム」の利用ないし 利用不能で生じた通常損害や特別損害、偶発損害、間接損害(データの消失や 不正確な処理、あなたが第三者が被った損失、あるいは「プログラム」が他のソフトウェアと一緒に動作しないという不具合などを含むがそれらに限らない)に一切の責任を負わない。そのような損害が生ずる可能性について彼らが忠告 されていたとしても同様である。

## 条件と制約終わり

### 以上の条項をあなたの新しいプログラムに適用する方法

あなたが新しいプログラムを開発したとして、公衆によってそれが利用される 可能性を最大にしたいなら、そのプログラムをこの契約書の条項に従って誰でも 再頒布あるいは変更できるようフリーソフトウェアにするのが最善です。

そのためには、プログラムに以下のような表示を添付してください。その場合、保証が排除されているということを最も効果的に伝えるために、それぞれの ソースファイルの冒頭に表示を添付すれば最も安全です。少なくとも、「著作権表示」という行と全文がある場所へのポインタだけは 各ファイルに含めて置いてください。

one line to give the program's name and an idea of what it does.  
Copyright (C) yyyy name of author

This program is free software; you can redistribute it and/or  
modify it under the terms of the GNU General Public License  
as published by the Free Software Foundation; either version 2  
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,  
but WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
GNU General Public License for more details.

You should have received a copy of the GNU General Public License  
along with this program; if not, write to the Free Software  
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

(訳)

*プログラムの名前と、それが何をするかについての簡単な説明。Copyright (C) 西暦年 作者の名前*

このプログラムはフリーソフトウェアです。あなたはこれを、  
フリーソフトウェア財団によって発行された GNU 一般公衆利用許諾契約書  
(バージョン2か、希望によってはそれ以降のバージョンのうちどれか)の  
定める条件の下で再頒布または改変することができます。

このプログラムは有用であることを願って頒布されますが、\*全くの無保証\*  
です。商業可能性の保証や特定の目的への適合性は、言外に示されたものも  
含め全く存在しません。詳しくはGNU 一般公衆利用許諾契約書をご覧ください。

あなたはこのプログラムと共に、GNU 一般公衆利用許諾契約書の複製物を一部  
受け取ったはずですが、もし受け取っていない場合は、フリーソフトウェア財団  
まで請求してください(宛先は the Free Software Foundation, Inc., 59  
Temple Place, Suite 330, Boston, MA 02111-1307 USA)。

電子ないし紙のメールであなたに問い合わせる方法についての情報も書き加えましょう。

プログラムが対話的なものならば、対話モードで起動した際に出力として 以下のような短い告知が表示されるようにしてください:

Gnomovision version 69, Copyright (C) year name of author  
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details  
type `show w'. This is free software, and you are welcome  
to redistribute it under certain conditions; type `show c'  
for details.

(訳)

Gnomovision バージョン 69, Copyright (C) 西暦年 作者の名前  
Gnomovision は\*全くの無保証\*で提供されます。詳しくは  
`show w' とタイプして下さい。  
これはフリーソフトウェアであり、ある条件の下で再頒布することが  
奨励されています。詳しくは `show c' とタイプして下さい。

ここで、仮想的なコマンド `show w' と `show c' は 一般公衆利用許諾契約書の適切な部分を表示するようになっていなければなりません。  
もちろん、あなたが使うコマンドを `show w' や `show c' と呼ぶ必然性はありませんので、あなたのプログラムに 合わせてマウスのクリックやメ  
ニューのアイテムにしても結構です。

また、あなたは、必要ならば(プログラマーとして働いていたら)あなたの 雇用主、あるいは場合によっては学校から、そのプログラムに関する「著作権放棄声明(copyright disclaimer)」に署名してもらうべきです。以下は例ですので、名前を変えてください:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the program `Gnomovision'
(which makes passes at compilers) written
by James Hacker.
```

signature of Ty Coon, 1 April 1989  
Ty Coon, President of Vice

(訳)

Yoyodyne社はここに、James Hackerによって書かれた  
プログラム `Gnomovision' (コンパイラへ通すプログラム)  
に関する一切の著作権の利益を放棄します。

Ty Coon氏の署名、1989年4月1日  
Ty Coon、副社長

この一般公衆利用許諾契約書では、あなたのプログラムを独占的なプログラムに 統合することを認めていません。あなたのプログラムがサブルーチンライブラリ ならば、独占的なアプリケーションとあなたのライブラリをリンクすることを 許可したほうがより便利であると考えられるかもしれません。もしこれがあなたの 望むことならば、この契約書の代わりに GNU ライブラリー一般公衆利用許諾契約書 [<http://www.fsf.org/licenses/lgpl.html>] を適用してください。

## GNU Free Documentation License

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.  
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

# 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

# 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member

of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties; any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and

legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O.Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

## 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the



aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

## ADDENDUM: How to use this License for your documents

Copyright (c) YEAR YOUR NAME.  
Permission is granted to copy, distribute and/or modify this document  
under the terms of the GNU Free Documentation License, Version 1.2  
or any later version published by the Free Software Foundation;  
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.  
A copy of the license is included in the section entitled "GNU  
Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the  
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

## GNU フリー文書利用許諾契約書

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA この利用許諾契約書を、一字一句そのままに複製し頒布することは許可する。しかし変更は認めない。

This is an unofficial translation of the GNU Free Documentation License into Japanese. It was not published by the Free Software Foundation, and does not legally state the distribution terms for documents that uses the GNU FDL--only the original English text of the GNU FDL does that. However, we hope that this translation will help Japanese speakers understand the GNU FDL better.

(訳: 以下はGNU Free Documentation Licenseの非公式な日本語訳です。これはフリーソフトウェア財団 (the Free Software Foundation)によって発表されたものではなく、GNU FDLを適用した文書の頒布条件を法的に有効な形で述べたものではありません。頒布条件としてはGNU FDLの英語版テキストで指定されているもののみが有効です。しかしながら、私たちはこの翻訳が、日本語を使用する人々にとってGNU FDLをより良く理解する助けとなることを望んでいます。)

# 0. はじめに

この利用許諾契約書の目的は、この契約書が適用されるマニュアルや教科書、その他機能本位で実用的な文書を(無料ではなく)自由という意味で「フリー」とすること、すなわち、改変の有無あるいは目的の営利非営利を問わず、文書を複製し再頒布する自由をすべての人々に効果的に保証することです。加えてこの契約書により、著者や出版者が自分たちの著作物に対して相応の敬意と賞賛を得る手段も保護されます。また、他人が行った改変に対して責任を負わずに済むようになります。

この利用許諾契約書は「コピーレフト」的なライセンスの一つであり、この契約書が適用された文書から派生した著作物は、それ自身もまた原本と同じ意味でフリーでなければなりません。この契約書は、フリーソフトウェアのために設計されたコピーレフトなライセンスであるGNU一般公衆利用許諾契約書を補足するものです。

(訳注: コピーレフト(copyleft)の概念については <http://www.gnu.org/copyleft/copyleft.ja.html> を参照せよ)

この利用許諾契約書は、フリーソフトウェア用のマニュアルに適用することを目的として書かれました。フリーソフトウェアはフリーな文書を必要としており、フリーなプログラムはそのソフトウェアが保証するのと同じ自由を提供するマニュアルと共に頒布されるべきだからです。しかし、この契約書の適用範囲はソフトウェアのマニュアルに留まりません。対象となる著作物において扱われる主題が何であれ、あるいはそれが印刷された書籍として出版されるか否かに関わらず、この契約書は文字で書かれたいかなる著作物にも適用することが可能です。私たちとしては、主にこの契約書を解説や参照を目的とする著作物に適用することをお勧めします。

# 1. この利用許諾契約書の適用範囲と用語の定義

著作物がこの利用許諾契約書の定める条件の下で頒布される旨の告知を、著作権者がその中に書いたすべてのマニュアルあるいはその他の著作物は、いかなる媒体上にあってもこの契約書の適用対象となる。そのような告知を置くことで、全世界において、著作権使用料を必要とせず、許可の存続期間を限定されること無く、この契約書の中で述べられている条件の下で当該著作物を利用できるという許可を与えることとする。以下において、「『文書』(Document)」とはそのような告知が記載されたマニュアルないし著作物すべてを指す。公衆の一員ならば誰でも契約の当事者となることができ、この契約書中では「あなた」と表現される。あなたは、著作権法の下で許可を必要とするような方法で著作物を複製や改変、あるいは頒布することにより、この契約書を受諾することになる。

『文書』の「改変版 (Modified Version)」とは、一字一句忠実に複製したが、あるいは改変や他言語への翻訳を行ったかどうかに関わらず、その『文書』の全体あるいは一部分を含む著作物すべてを意味する。

「補遺部分 (Secondary Section)」とは、『文書』中でその旨指定された補遺ないし本文に先だって前付けとして置かれる一部分であり、『文書』の出版者あるいは著者と、『文書』全体の主題 (あるいはそれに関連する事柄)との関係のみを論じ、全体としての主題の範疇に直接属する内容を

全く含まないものである (たとえば、『文書』の一部が数学の教科書だった場合、補遺部分では数学について何も解説してはならない)。補遺部分で扱われる関係は、その主題あるいは関連する事柄との歴史的なつながりのことかも知れないし、それらに関する法的、商業的、哲学的、倫理的、あるいは政治的立場についてかも知れない。

「変更不可部分 (Invariant Sections)」とは補遺部分の一種で、それらが変更不可部分であることが、『文書』をこの利用許諾契約書の下で発表する旨述べた告知中においてその部分の題名と共に明示されているものである。ある部分が上記のような「補遺」性の定義にそぐわない場合は、その部分を「変更不可」として指定することは認められない。『文書』は、変更不可部分を全く含まなくても良い。『文書』において変更不可部分が全く指定されていないければ、その『文書』に変更不可部分は存在しないということである。

「カバーテキスト (Cover Texts)」とは、『文書』がこの利用許諾契約書の指定する条件の下で発表される旨述べた告知において、「表カバーテキスト」あるいは「裏カバーテキスト」として列挙された短い文章のことを指す。表カバーテキストは最大で5語、裏カバーテキストは最大で25語までとする。

『文書』の「透過的」複製物とは、機械による読み取りが可能な『文書』の複製物のことを指す。透過的な複製物の文書形式は、その仕様が一般の人々に入手可能で、『文書』の内容を一般的なテキストエディタ、または(画素で構成される画像ならば)一般的なペイントプログラム、あるいは(図面ならば)いくつかの広く入手可能な製図エディタで簡単に改訂するのに適しており、なおかつテキストフォーマットへの入力に適する(あるいはテキストフォーマットへの入力に適する諸形式への自動的な変換に適する)ものでなければならない。透過的なファイル形式への複製であっても、マークアップ、あるいはマークアップの不在が読者によるそれ以降の改変をわざと邪魔し阻害するように仕組まれたものは透過的であるとは見做されない。ある画像形式が、相当量のテキスト文章を表現するために使われた場合、それは透過的ではない。透過的ではない複製は「非透過的」複製と呼ばれる。

透過的複製に適した形式の例としては、マークアップを含まないプレーンな ASCII 形式、Texinfo 入力形式、LaTeX 入力形式、一般に入手可能な DTD を用いた SGML あるいは XML、または人間による改変を想定して設計された、標準に準拠したシンプルなもの HTML や PostScript、PDF などが挙げられる。透過的な画像形式の例には、PNG や XCF、JPG が含まれる。非透過な形式としては、独占的なワードプロセッサでのみ閲覧編集できる独占的なファイル形式、普通には入手できない DTD または処理系を使った SGML や XML、ある種のワードプロセッサが生成する、出力のみを目的とした機械生成の HTML や PostScript、PDF などが含まれる。

「題扉 (Title Page)」とは、印刷された書籍に於いては、実際の表紙自身のみならず、この利用許諾契約書が表紙に掲載することを義務づける文章や図などを、読みやすい形で載せるのに必要なだけの、表紙に引き続き数ページをも意味する。表紙に類するものが無い形式で発表される著作物においては、「題扉」とは本文の始まりに先だって、その著作物の題名が最も目立つ形で現れる場所の近くに置かれる文章のことを指す。

「XYZ」と題された (Entitled XYZ) 部分とは、『文書』において「XYZ」と名付けられた一部分であり、その題名は正確に「XYZ」であるか、「XYZ」を他の言語に翻訳した上でその後ろに「XYZ」をそのまま括弧で括ったものを含む記述のどちらかである(ここでの「XYZ」とは、この利用許諾契約書において以下で言及される特定の部分名を意味している。例えば「謝辞 (Acknowledgements)」、「献辞 (Dedications)」、「推薦の辞 (Endorsements)」、「履歴 (History)」)。あなたが『文書』を改変する場合、そのような部分の「題名を保存する (Preserve the Title)」とは、「XYZ」と題された」部分として、ここでの定義に従い題名を残すということである。

『文書』は、「保証否認警告 (Warranty Disclaimers)」を、この利用許諾契約書が『文書』に適用されると述べた告知の次に含んでも良い。この種の保証否認警告は、この契約書からの言及という形で利用条件に含まれるものと解されるが、保証の否認に関することについてののみ有効とする。こういった保証否認警告で示うその他のいかなる含意も無効であり、この契約書の効能には何ら影響を持たない。

## 2. 逐語的に忠実な複製

この利用許諾契約書、著作権表示、この契約書が『文書』に適用される旨述べた告知の三つがすべての複製物に複製され、かつあなたがこの契約書で指定されている以外のいかなる条件も追加しない限り、あなたはこの『文書』を、商用であるか否かを問わずいかなる形で複製頒布することができる。あなたは、あなたが作成あるいは頒布する複製物に対して、閲覧や再複製を技術的な手法によって妨害、規制してはならない。しかしながら、複製と引き換えに代価を得てもかまわない。あなたが相当量の複製物を頒布する際には、本契約書第3項で指定される条件にも従わなければならない。

またあなたは、上記と同じ条件の下で、複製物を貸与したり複製物を公に開示することができる。

## 3. 大量の複製

もしあなたが、『文書』の印刷された (あるいは通常は印刷された表紙を持つ媒体における)複製物を100部を超えて出版し、また『文書』の利用許諾告知がカバーテキストの掲載を要求している場合には、指定されたすべてのカバーテキストを、表カバーテキストは表表紙に、裏カバーテキストは裏表紙に、はっきりと読みやすい形で載せた表紙の中に複製物本体を綴じ込まなければならない。また、両方の表紙において、それらの複製物の出版者としてのあなたをはっきりとかつ読みやすい形で確認できなければならない。表表紙では『文書』の完全な題名を、題名を構成するすべての語が等しく目立つようにして、視認可能な形で示さなければならない。それらの情報に加えて、表紙に他の文章や図などを加えることは許可される。表紙のみを変更した複製物は、それが『文書』の題名を保存し上記の条件を満たす限り、ほかの点では逐語的に忠実な複製物として扱われる。

もしどちらかの表紙に要求されるカバーテキストの量が多すぎて読みやすく収めることが不可能ならば、あなたはテキスト先頭の一文(あるいは適切に収まるだけ)を実際の表紙に載せ、続きは隣接したページに載せるべきである。

あなたが『文書』の「非透過的」複製物を100部を超えて出版あるいは頒布する場合、それぞれの非透過な複製物と一緒に機械で読み取り可能な透過的複製物を添付するか、それぞれの非透過な複製物(あるいはそれに付属する文書)中で、公にアクセス可能なコンピュータネットワーク上の所在地を記述しなければならない。その場所には、非透過な複製物と内容的に寸分違わず、余計なものが追加されていない完全な『文書』の透過的複製物が置かれ、またそこから、ネットワークを利用する一般公衆が、一般に標準的と考えられるネットワークプロトコルを使ってダウンロードすることができなければならない。もしあなたが後者の選択肢を選ぶならば、その版の非透過な複製物を公衆に(直接、あるいはあなたの代理人ないし小売業者が)最後に頒布してから最低1年間は、その透過的複製物が指定の場所でアクセス可能であり続けることを保証するよう、非透過な複製物の大量頒布を始める際に十分に慎重な手順を踏まなければならない。

これは要望であり必要条件ではないが、『文書』の著者に、『文書』の更新された版をあなたに提供する機会を与えるため、透過非透過を問わず大量の複製物を再頒布し始める前には彼らにきちんと連絡しておいてほしい。

## 4. 改変

『文書』の改変版を、この利用許諾契約書と細部まで同一の契約の下で発表する限り、すなわち原本の役割を改変版で置き換えた形での頒布と改変を、その複製物を所有するすべての人々に許可する限り、あなたは改変版を上記第2項および第3項が指定する条件の下で複製および頒布することができる。さらに、あなたは改変版において以下のことを行わなければならない。

- A. 題扉に(もしあればその他の表紙にも)、『文書』および『文書』のそれ以前の版と見分けがつく題名を載せること(もし以前の 版があれば、『文書』の「履歴 (History)」の部分に列記されているはずで ある)。もし元の版の出版者から許可を得たならば、以前の版と同じ題名を使ってもいい。
- B. 題扉に、改変版における改変を行った1人以上の人物 が団体名を列記すること。あわせて元の『文書』の著者として、最低5人(もし5人以下ならばすべて)の主要著者を列記すること。ただし元の著者たちが この条件を免除した場合は除く。
- C. 題扉に、改変版の出版者名を出版者として記載すること。
- D. 『文書』にあるすべての著作権表示を残すこと。
- E. 他の著作権表示の近くに、あなたの改変に対する適 当な著作権表示を追加すること。
- F. 著作権表示のすぐ後に、改変版をこの契約書の条件 の下で利用することを公衆に対して許可する告知を含めること。その形式は この契約書の末尾にある付記で示されている。
- G. 元の『文書』の利用許諾告知に書かれた、変更不可 部分の完全な一覧と、要求されるカパーテキストとを、改変版の利用許諾告知 中でもそのまま残すこと。
- H. この契約書の、変更されていない複製物を含めること。
- I. 「履歴 (History)」と題された部分とその題名を保存し、そこに改変版の、少なくとも題名、出版年、新しく変更した部分の著 者名、出版者名を、題扉に掲載するのと同じように記載した一項を加えること。もし『文書』中に「履歴」と題された部分が存在しない場合には、『文 書』の題名、出版年、著者、出版者を題扉に掲載するのと同じように記載した部分を用意し、上記で述べたような、改変版を説明する一項を加えること。
- J. 『文書』中に、『文書』の透過的複製物への公共的 アクセスのために指定されたネットワークの所在地が記載されていたならば、それを保存すること。同様に、その『文書』の元になった以前の版で指定 されていたネットワークの所在地も載っていたならば、それも保存すること。これらの情報は「履歴(History)」の部分に置いても良い。ただし、それが『文書』自身より少なくとも4年前に出版された著作物の情報であったり、あるいは改変版が参考になっている版の元の出版者から許可を得たならば、その情報を削除してもかまわない。
- K. 「謝辞 (Acknowledgement)」あるいは「献辞 (Dedication)」等と題されたいかなる部分も、その部分の題名を保存し、そ の部分の内容(各貢献者への謝意あるいは献呈の意)と語調を保存すること。
- L. 『文書』の変更不可部分を、その本文および題名を 変更せずに保存すること。章番号やそれに相当するものは部分の題名の一 部とは見做さない。
- M. 「推薦の辞 (Endorsement)」というような章名が題 された部分はすべて削除すること。そのような部分を改変版に含めてはならない。
- N. すでに存在する部分を「推薦の辞 (Endorsement)」と題されるように改名したり、題名の点で変更不可部分のどれかと衝突する ように改名してはならない。
- O. 保証否認警告を保存すること。

もし改変版に、補遺部分としての条件を満たし、かつ『文書』から複製物された文章や図などをいっさい含んでいない、前書き的な章あるいは付録が新しく含まれるならば、あなたは希望によりそれらの部分の一部あるいはすべてを変更不可と宣言することができる。変更不可を宣言するためには、それらの部分の題名を改変版の利用許諾告知中の変更不可部分一覧に追加すれば良い。これらの題名は他の章名とは全く別のものでなければならない。

含まれる内容が、さまざまな集団によるあなたの改変版に対する推薦の辞のみである限り、あなたは、「推薦の辞 (Endorsement)」と題された章を追加することができる。推薦の辞の例としては、ピアレビューの陳述、あるいは文書がある標準の権威ある定義としてその団体に承認されたという声明などがある。

あなたは、5語までの一文を表カバーテキストとして、25語までの文を表裏紙テキストとして、改変版のカバーテキスト一覧の末尾に加えることができる。一個人ないし一団体が直接(あるいは団体内で結ばれた協定によって)加えることができるのは、表カバーテキストおよび裏カバーテキストとしてそれぞれ一文ずつのみである。もし以前すでにその文書において、表裏いずれかの表紙にあなたの(またはあなたが代表する同じ団体内で為された協定に基づく)カバーテキストが含まれていたならば、あなたが新たに追加することはできない。しかしあなたは、その古い文を加えた以前の出版者から明示的な許可を得たならば、古い文を置き換えることができる。

『文書』の著者あるいは出版者は、この利用許諾契約書によって、彼らの名前を利用することを許可しているわけではない。彼らの名前を改変版の宣伝に使ったり、改変版への明示的あるいは黙示的な保証のために使うことを許可するものではない。

## 5. 文書の結合

あなたは、上記第4項において改変版に関して定義された条件の下で、この利用許諾契約書の下で発表された複数の文書の一つにまとめることができる。その際、原本となる文書にある変更不可部分を全て、改変せずに結合後の著作物中に含め、それらをあなたが統合した著作物の変更不可部分としてその利用許諾告知において列記し、かつ原本にある全ての保証否認警告を保存しなければならない。

結合後の著作物についてはこの契約書の複製物一つ含んでいれよく、同一内容の変更不可部分が複数ある場合には一つで代用してよい。もし同じ題名だが内容の異なる変更不可部分が複数あるならば、そのような部分のそれぞれの題名の最後に、(もし分かっているならば)その部分の原著者あるいは出版者の名前で、あるいは他と重ならないような番号を括弧で括って記載することで、それぞれ見分けが付くようにしなければならない。結合後の著作物の利用許諾告知における変更不可部分の一覧においても、章の題名に同様の調整をすること。

結合後の著作物においては、あなたはそれぞれの原本の「履歴 (History)」と題されたあらゆる部分をまとめて、「履歴 (History)」と題された一章にしなければならない。同様に、「謝辞 (Acknowledgements)」あるいは「献辞 (Dedications)」と題されたあらゆる部分もまとめなければならない。あなたは「推薦の辞 (Endorsements)」と題されたあらゆる部分も削除しなければならない。

## 6. 文書の収集

あなたは、この利用許諾契約書の下で発表された複数の文書で構成される収集著作物を作ることができる。その場合、それぞれの文書が逐語的に忠実に複製されることを保障するために他のすべての点でこの契約書の定める条件に従う限り、さまざまな文書中のこの契約書の個々の複製物を、収集著作物中に複製物一つ含めることで代用することができる。

あなたは、このような収集著作物から文書一つ取り出し、それをこの契約書の下で頒布することができる。ただしその際には、この契約書の複製物を抽出された文書に挿入し、またその他のすべての点でこの文書の逐語的に忠実な複製に関してこの契約書が定める条件に従わなければならない。

## 7. 独立した著作物の集積

『文書』あるいはその派生物を、他の別の独立した文書あるいは著作物と一緒にし、一巻の記憶装置あるいは頒布媒体に収めた編集著作物は、編集に起因する著作権が編集著作物に含まれる個々の著作物がその利用者に許可した法的権利を制限するよう行使されない限り、「集積」著作物と呼ばれる。『文書』が集積著作物に含まれる場合、この契約書は、『文書』と共にまとめられた他の独立した著作物には、それら自身が『文書』の派生物で無い限り適用されることにはならない。

このような『文書』の複製物において、この利用許諾契約書の第3項によりカバーテキストの掲載が要求されている場合、『文書』の量が集積著作物全体の2分の1以下であれば、『文書』のカバーテキストは集積著作物中で『文書』そのものの周りを囲む中表紙、あるいは『文書』が電子的形式である場合には表紙の電子的等価物にのみ配置するだけでよい。その場合以外は、カバーテキストは集積著作物全体を取り巻く印刷された表紙に掲載されなければならない。

## 8. 翻訳

翻訳は改変の一種と見做すので、あなたは『文書』の翻訳をこの利用許諾契約書の第4項の定める条件の下で頒布することができる。変更不可部分を翻訳によって置き換えるには著作権者の特別許可を必要とするが、元の変更不可部分に追加する形で変更不可部分の全てないし一部の翻訳を含めることはかまわない。この契約書や『文書』中の利用許諾告知、保証否認警告すべての英語原本も含める限り、あなたはこの契約書、告知、警告の翻訳を含めることができる。契約書や告知、警告に関して翻訳と英語原本との間に食い違いが生じた場合、英語原本が優先される。

典型的な例として、『文書』のある部分が原文で「Acknowledgements」、「Dedications」、あるいは「History」と題されていた場合、実際の題名を変更するには、題名を保存する(この契約書の第1項)ための条件(同第4項)を満たすことが必要となる。

## 9. 契約の終了

この利用許諾契約書の下で明確に提示されている場合を除き、あなたは『文書』を複製、改変、サブライセンス、あるいは頒布してはならない。このライセンスで指定されている以外の、『文書』の複製、改変、サブライセンス、頒布に関するすべての企ては無効であり、この契約書によって保証されるあなたの権利を自動的に終結させることとなる。しかし、この契約書の下であなたから複製物ないし諸権利を得た個人や団体に関しては、そういった人々がこの契約書に完全に従ったままである限り、彼らに与えられた許諾は終結しない。

## 10. 将来における本利用許諾契約書の改訂

フリーソフトウェア財団は、時によってGNU フリー文書利用許諾契約書の新しい改訂版を出版することができる。そのような新版は現在の版と理念においては似たものになるであろうが、新たに生じた問題や懸念を解決するため細部においては違ったものになるだろう。詳しくは <http://www.gnu.org/copyleft/> を参照せよ。

GNU フリー文書利用許諾契約書のそれぞれの版には、新旧の区別が付くようなバージョン番号が振られている。もし『文書』において、この契約書のある特定の版が「それ以降のどの版でも」適用して良いと指定されている場合、あなたはフリーソフトウェア財団から発行された(草稿として発表されたものを除く)指定の版かそれ以降の版のうちどれか一つを選び、その条項や条件に従うことができる。もし『文書』がこの契約書のバージョン番号を指定していない場合には、あなたはフリーソフトウェア財団から今までに出版された(草稿として発表されたものを除く)版のうちからどれか一つを選ぶことができる。

## 付録: この利用許諾契約書をあなたの文書に適用するには

Copyright (c) YEAR YOUR NAME.  
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

(訳:

Copyright (C) 西暦年 あなたの名前.

この文書を、フリーソフトウェア財団発行の GNU フリー文書利用許諾契約書(バージョン1.2かそれ以降から一つを選択)が定める条件の下で複製、頒布、あるいは改変することを許可する。変更不可部分、表カバーテキスト、裏カバーテキストは存在しない。この利用許諾契約書の複製物は「GNU フリー文書利用許諾契約書」という章に含まれている。

)

もし変更不可部分や表カバーテキスト、裏カバーテキストがあれば、「変更不可部分…は存在しない。」というところを以下で置き換えてください:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

(訳:

(章の題名を列記)は変更不可部分であり、(表カバーテキストを列記)は表カバーテキスト、(裏カバーテキストを列記)は裏カバーテキストである。

)

変更不可部分はあるがカバーテキストは存在しないなど、その他の三者の組み合わせに関しては、状況に合わせて上記二つの選択肢を混ぜてください。

あなたの文書に、他に類を見ない独自のプログラムコードのサンプルが含まれる場合、フリーソフトウェアにおいてそのコードを利用することを許可するために、そういったサンプルに関してはこの利用許諾契約書と同時にGNU 一般公衆許諾契約書のようなフリーソフトウェア向けライセンスのうちどれか一つを選択して適用してもよい、というような条件の下で発表することを推奨します。

